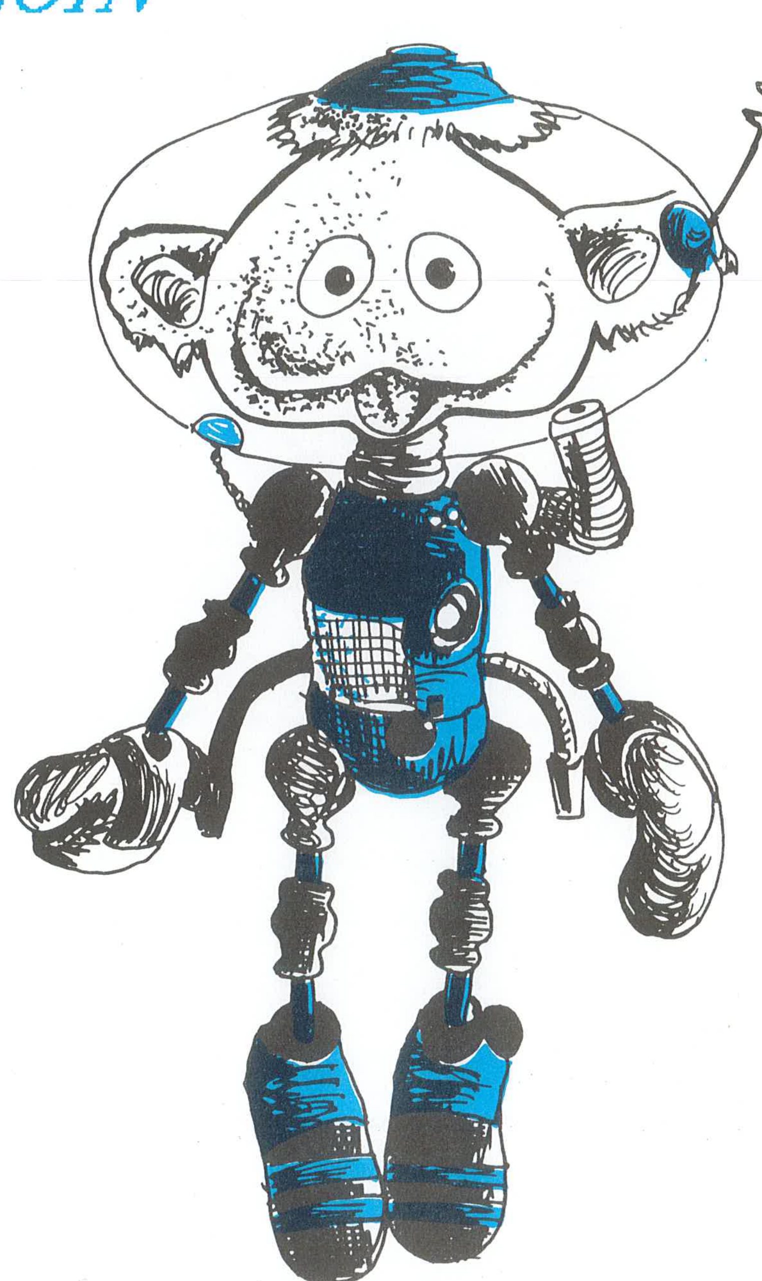


12

TURBO-Forth: le virus du programmeur

FEVRIER 1988





POUR
TURBO-FORTH

EDITORIAL

Avis aux adhérents qui voudraient nous laisser tomber: JEDI prépare derrière ses fagots des nouveautés pour TURBO-Forth absolument révolutionnaires.

La première, et pas la moindre, un méta-compilateur avec routines de compactage automatique du code généré lors d'une méta-compilation. Un outil de cette qualité n'a encore JAMAIS été diffusé par une revue. Mais nous n'en dirons pas plus pour le moment. Diffusion prévue dans deux mois.

La seconde, JEDI peut maintenant, à l'aide de son nouveau service télématique, vous assister dans tous les problèmes techniques de programmation que vous avez à résoudre en exploitant TURBO-Forth. C'est le "service après vente" attaché à TURBO-Forth.

La troisième, les programmes diffusés dans JEDI sont maintenant téléchargeables. Une routine vous plait? Ne la recopiez plus. Branchez-vous sur 3615 SAM*JEDI et téléchargez le programme. Ensuite, compilez-le. Le serveur SAM*JEDI, votre nouveau média pour aller encore plus vite dans l'acquisition de TURBO-Forth.

Et de plus, toutes ces nouveautés ne sont pas réservées aux seuls PC. Bientôt, le téléchargement sera opérationnel sur les systèmes ATARI, APPLE et tout système dont vous nous fournirez la matière première: vos trucs, vos astuces et vos prorammes.

Alors bientôt on pourra affirmer qu'il y a deux génération de TURBOs, celle des TURBO-PASCAL-PROLOG-C-BASIC-etc... et la GENERATION TURBO-Forth.

SOMMAIRE

FORTH: DU NOUVEAU POUR Volks-FORTH SUR ATARI — 2
un FORTH made in GERMANY qui n'est pas passé par la Belgique...

LA VIRGULE FLOTTANTE 83-STANDARD — 5
des routines facilement adaptables à d'autres systèmes que TURBO-Forth

ROUTINES DE CALCUL EN QUADRUPLE PRECISION — 10
pour des calculs très précis. Réservés aux FORTH-maniaques parfois insomniaques.

PETIT UTILITAIRE Volks-FORTH: WHEREIS — 13
pour que Volks-FORTH ne reste pas dans votre tiroir.

PETITES MODIFICATIONS DE TURBO-FORTH — 18
pour méta-compiler en extra-segment

OCCAM: UN LANGAGE POUR LES ORDINATEURS DE LA PROCHAINE GENERATION — 14
si vous êtes un drogué de micro, l'OCCAM devrait vous faire de l'effet.

PROLOG: CALCUL DE RESISTANCES — 17
pour les branchés en électronique

TELEMATIQUE:
JEDI SUR 3615 — 19
tout ce qu'il faut savoir pour être à la page.

CONNEXION AU RESEAU BTX (BILDSCHIRMTEXT) — 20
Oui! Maintenant on peut consulter l'annuaire électronique allemand depuis son MINITEL en France sans passer par le 19! Je connais des frontaliers, en Alsace et en Lorraine, que ça intéressera beaucoup. Mais les germanophones de France et de Navarre ne seront pas en reste. L'Europe, mon cher, l'Euro-télématique...

Nos coordonnées: ASSOCIATION JEDI, 17, rue de la Lancette 75012 PARIS
tel président: (1) 43.40.96.53
tel secrétaire: (1) 46.56.33.67 ou 36.15 SAM*JEDI et bal: SECRETAIRE
télématique: 36.15 SAM*JEDI

Le serveur SAM est la propriété de la Sté VICTEL, 14 bd Montmartre 75009 PARIS, tél: 45.23.02.79

FORTH

DU NOUVEAU POUR Volks-FORTH SUR ATARI

par Daniel FLIPO

Volks-FORTH sur ATARI (83-Standard)

Enfin des nouvelles de Volks-FORTH 83-Standard pour ATARI:

- un éditeur complètement francisé (menu et boîtes de dialogue)
- les commandes CTRL A, CTRL M, CTRL W de cet éditeur fonctionnent maintenant normalement (les claviers farnçais et allemand étant différents, une adaptation s'imposait...)
- Un fichier ASCII LISEZ.MOI qui contient les principales explications nécessaires à l'utilisation de ce FORTH.

Rappelons que le n° 35 de JEDI contenait (pages 17 à 20) déjà une présentation de ce langage et les premières indications pour la mise en route...

Les adhérents qui ont acquis l'ancienne version toute en allemand pourront obtenir la nouvelle version en envoyant:

- une enveloppe (matelassée de préférence) timbrée à 5,60 Fr portant leur adresse,
- une disquette 3' vierge à l'adresse suivante:

Daniel FLIPO
1, bis rue St Jacques
59800 LILLE

Ils recevront leur disquette contenant les fichiers modifiés (EDITOR.SCR, EDIICON.RSC, EDIICON.SCR, PRINTER.SCR, STARTUP.SCR et LISEZ.MOI) et la nouvelle version de 4TH.PRG.

Il reste encore beaucoup de travail à faire (avis aux courageux qui lisent l'allemand!) pour traduire la documentation imposante et très bien faite qui accompagne ce FORTH, mais j'espère que cette nouvelle présentation permettra à un plus grand nombre d'Ataristes de profiter des remarquables possibilités de ce langage très bien adapté à leur machine, qui offre:

- un éditeur pleine page entièrement sous GEM (pas besoin d'attendre TURBO-Forth!) avec fonctions de remplacement de chaînes de caractères, recopie de morceaux d'écrans sur d'autres, etc..., toutes ces fonctionnalités étant accessibles à la fois par menu déroulant ou directement au clavier.

- un traceur très commode pour "débugger" les programmes récalcitrants.

- une possibilité de travail en multi-tâches autorisant par exemple l'impression d'un listing pendant la compilation ou l'exécution d'un autre programme.

- un interface permettant de développer une application sous GEM avec menus déroulants, bôites de dialogue, graphismes, etc... toutes les fonctions GEM du C (Mégamax, Lattice) sont implantées avec en plus des macros très pratiques d'emploi (affichage d'une boîte de dialogue avec sauvegarde de l'écran avant ouverture et restitution après fermeture, en une seule instruction par exemple...).

J'espère vous avoir suffisamment mis l'eau à la bouche et vous souhaite bien du plaisir!

I) EDITEUR SOUS GEM

Après avoir double-cliqué 4TH.PRG pour entrer sous FORTH, vous pouvez:

1) CHOISIR UN FICHIER EXISTANT COMME FICHIER COURANT en tapant

USE NOM_FICHIER.SCR <return>

puis 3 L <return> pour entrer sous éditeur avec l'écran 3 de ce fichier dans la fenêtre. Pour connaître le nombre d'écrans du fichier courant, il suffit de taper (sous FORTH) CAPACITY . <return>. Les écrans sont numérotés de 0 à "capacity-1", et l'écran 0 ne peut contenir que des commentaires (description du fichier...) car il n'est jamais compilé.

Sous éditeur, essayez à loisir les différentes options du menu (protégez vos disquettes avant!) pour vous familiariser avec ses nombreuses possibilités. A l'appel de chaque option du menu apparaît une boîte de dialogue qui vous renseigne sur l'effet de l'option choisie, et vous permet d'annuler ou de confirmer votre choix. En face de chaque option du menu figure le code clavier équivalent à la fonction appelée: lorsque vous connaîtrez mieux l'éditeur, vous pourrez ainsi ne plus utiliser la souris, en tapant par exemple CTRL N pour passer à l'écran suivant (CTRL=touche Control désignée par ^ dans le menu, et N pour Next), ou CTRL B pour revenir à l'écran précédent.

Pour sortir de l'éditeur, on a le choix entre 4 options (menu SORTIES) dont les deux plus usuelles sont:

ESC (touche escape) pour sortir sans rien sauvegarder
CTRL S (S pour SAVE) pour sauvegarder sur disquette toutes les modifications faites.

Après être sorti de l'éditeur, on peut rappeler le dernier écran vu par V <return> (ou bien sûr son numéro suivi de L puis de <return>). Une autre utilisation de V: lors d'un arrêt en cours de compilation (erreur...?) il suffit de faire V <return> pour revenir sous éditeur, le curseur étant placé automatiquement juste derrière le mot qui a provoqué l'erreur.

2) CREER UN NOUVEAU FICHIER SOURCE

Taper pour cela sous FORTH

MAKEFILE NOM_FICHIER.SCR <return>

puis 2 MORE pour allouer deux écrans (de 16 lignes de 64 caractères, soit 1024 octets) dont les numéros sont 0 et 1.

Par la suite on pourra ajouter d'autres écrans si nécessaire en tapant par exemple 3 MORE pour ajouter trois écrans au fichier, mais il est très peu commode d'en supprimer...

3) COMPILER UN FICHIER SOURCE.

Pour compiler un fichier en entier (c'est à dire tous les écrans sauf 0) on tape sous FORTH:

INCLUDE NOM_FICHIER.SCR <return>

Si l'on ne veut compiler que l'écran 2 du fichier courant (chargé par USE NOM_FICHIER.SCR) on fait 2 LOAD <return> ou 3 8 THRU <return> pour compiler les écrans 3 à 8 inclus.

On peut bien sûr supprimer des mots du dictionnaire grâce à FORGET. En règle générale, ce FORTH suit le standard F83. Pour vous en convaince, listez les différents vocabulaires par WORDS: ONLYFORTH WORDS <return> listera le noyau FORTH... Notez également que l'interpréteur ne se soucie pas des majuscules: allot ALLOT ou aLLoT ont le même effet...

4) RECOPIER UNE DEFINITION D'UN FICHIER DANS UN AUTRE

Sous FORTH, VIEW MOT <return> fait passer sous éditeur avec le curseur placé juste derrière le mot MOT, ce qui permet de consulter sa définition. Le fichier courant est alors le fichier contenant MOT. Le nom de ce fichier et le numéro d'écran sont affichés dans la fenêtre.

Exemple: SOURCE.SCR écran 6

On peut copier tout cet écran sur l'écran 3 du fichier DEST.SCR en tapant (sous FORTH, sortir de l'éditeur par ESC)

DEST.SCR 6 FROM SOURCE.SCR 3 COPY <return>

De même pour copier les écrans 5 à 8 inclus du fichier SOURCE sur les écrans 2 et suivants du fichier DEST, faire

DEST.SCR 5 8 FROM SOURCE.SCR 2 CONVEY <return>

S'il s'agit de recopier des écrans à l'intérieur d'un même fichier (le fichier courant), il suffit de taper

6 3 COPY <return> pour copier l'écran 6 sur l'écran 3, ou
5 8 7 CONVEY <return> pour copier les écrans 5 à 8 à partir de l'écran 7 (les zones sources et destination peuvent se chevaucher sans problème...).

On peut également copier seulement qeulques lignes d'un écran en restant sous éditeur: l'option VOIR MOT du menu (ou son équivalent CTRL V) permet de rechercher une définition dans l'ensemble des fichiers .SCR. On sort de la boîte de dialogue par MARQUER (pour pouvoir revenir à l'écran actuel où la définition de MOT doit être insérée) et on recopie les lignes désirées en plaçant le curseur dans la première ligne à copier, puis en tapant CTRL down (flèche vers bas du pavé curseur) autant de fois que de lignes à copier (ces lignes sont stockées dans un tampon spécial), on fait ensuite CTRL W pour revenir à l'écran initial, on place le curseur à l'endroit choisi pour l'insertion, et on recopie les lignes du tampon par SHIFT down.

5) ENCORE QUELQUES INDICATIONS SUR L'EDITEUR...

Le mot \ interdit la compilation du texte qui suit sur la même ligne (utilisé pour les commentaires...), et le mot \\ fait de même sur la fin d'écran.

Le mot \needs sert à la compilation conditionnelle d'écrans. Exemples:

```
\needs code     include assemble.scr
\needs >absaddr : >absaddr forthstart 0 D+ ;
```

Si le mot qui suit \needs figure déjà dans le dictionnaire (ou plus précisément dans les vocabulaires autorisés en lecture), le compilateur ignore la fin de ligne et passe à la suivante, sinon il exécute la fin de ligne. Ceci permet d'inclure les définitions de mot nécessaires seulement si elles ne figurent pas déja dans le vocabulaire.

II) LE TRACEUR (inclus dans TOOLS.SCR)

Les mots dont la définition commence par : peuvent tous être exécutés pas à pas en mode "trace". Pour tester le mot MOT, on fait: DEBUG MOT <return> pour lancer le mode "trace", puis on exécute MOT (ou tout autre mot contenant MOT dans sa définition), qautres colonnes apparaissent alors à l'écran:

Adresse(k)   CFA(k)   MOT(k)   Etat de la pile

- MOT(k) est le kième mot de la définition de MOT. Il sera exécuté à la prochaine frappe de <return>
- Adresse(k) est l'adresse dans le dictionnaire de MOT(k) en tant que partie de MOT (exprimée en hexadécimal)
- CFA(k) est l'adresse de compilation de MOT(k) (en hexa)
- Etat de la pile est vu avant l'exécution de MOT(k), le sommet de la pile étant le terme le plus à gauche... Les valeurs sont exprimées dans la base courante.

Chaque frappe de <return> provoque l'exécution du mot MOT(k) affiché dans la dernière ligne. On peut arrêter l'exécution de MOT à tout moment en frappant

RESTART <return>

On peut également rentrer plus profondément en mode trace pour faire exécuter pas à pas le mot MOT(K) inclus dans la partie de définition de MOT, pour cela taper

NEST <return>

à la fin de MOT(k) on continuera à "tracer" MOT...

Encore une possibilité remarquable: si MOT contient une boucle qui doit être exécutée un grand nombre de fois, il arrive qu'après avoir vérifié une ou deux exécutions de celle-ci on souhaite parcourir les suivantes en temps réel. Pour cela taper

ENDLOOP <return>

à la fin de la boucle. Le traceur s'arrêtera ainsi à chaque parcours de la boucle à la ligne où figure ENDLOOP.

Au cours d'exécution d'un mot en mode "trace", vous avez à chaque ligne la possibilité de faire exécuter les mots FORTH de votre choix avant MOT(k): vous pouvez par exemple ajouter des paramètres sur la pile, en enlever, faire des SWAP ROT etc..., ou visualiser le contenu de variables (taper NOM_VARIABLE @ .) ou d'une zone mémoire:

adr_début(16bits) longueur(16bits) DUMP

pour une zone située à l'intérieur des 64 Ko du FORTH, ou

adr_début(32bits) longueur(16bits) LDUMP

pour une zone définie par une adresse absolue.

On quitte le mode "trace" par RESTART ou par END-TRACE. Essayez le fonctionnement de ce remarquable outil sur quelques exemples, il vous permettra de localiser vite et bien les (rares?) erreurs que vous auriez pu commettre...

III) L'IMPRESSION DES FICHIERS

Tous les mots provoquant des affichages sont vectorisés (EMIT TYPE SPACE PAGE AT AT? etc...).

PRINT provoque la sortie sur imprimante, tandis que DISPLAY revient à l'affichage sur écran.

1) Les principaux mots à connaître sont les suivants:

PRINTALL imprime tout le fichier courant

Exemple: si on tape USE ASSEMBLE.SCR PRINTALL <return>
la totalité du fichier ASSEMBLE.SCR (y compris l'écran 0) est imprimée à raison de 6 écrans par page en deux colonnes de trois écrans (format condensé).

LISTING imprime tout le fichier courant, avec l'écran commentaire (s'il existe) à droite de l'écran source qu'il commente.

Exemple: USE PRINTER.SCR LISTING <return>

Utile pour imprimer les sources commentés: EDITOR.SCR, EDWINDOW.SCR, SUPERGEM.SCR, RAMDISK.SCR...

PTHRU 2 5 PTHRU <return> imprime les écrans 2, 3, 4 et 5 du fichier courant sur une même page:

Ecran 2   Ecran 4
Ecran 3   Ecran 5

DOCUMENT comme PTHRU mais avec écrans commentaires.

Exemple: si le fichier courant est PRINTER.SCR,

11 13 DOCUMENT <return>

imprime les écrans 11 à 13 et les écrans commentaires associés:

Ecran 11 Ecran 26
Ecran 12 Ecran 27
Ecran 13 Ecran 28

2) Réglages:

L'interface PRINTER.SCR est prévue pour les imprimantes compatibles EPSON. Pour adapter d'autres types d'imprimantes, il suffit de s'inspirer des écrans 4 et 5 (commentés 19 et 20) et de la documentation de l'imprimante.

Pour nous français, l'écran 6 ne sert à rien, il vaut mieux ne pas le compiler (d'où le \\ au début de cet écran).

La longueur du papier a été réglée à 11 pouces (environ 28 cm), on peut la porter à 12 pouces en remplaçant &11 par &12 dans la définition de NORMAL (écran 5 de PRINTER.SCR).

3) Impression en arrire-plan.

La compilation du fichier TASKER.SCR avant celle de PRINTER.SCR permet de disposer du mot SPOLL' qui installe un spooler d'imprimante. On l'utilise de la manière suivante:

- taper MULTITASK <return> pour mettre le système en fonctionnement multi-tâches (on revient en monotâche en tapant SINGLETASK <return>)
- puis SPOOL' PRINTALL <return>
  ou SPOOL' LISTING <return>
  ou 2 5 SPOOL' PTHRU <return>
  ou 11 13 SPOOL' DOCUMENT <return>

pour reprendre les mêmes exemples que ci-dessus mais en récupérant la main immédiatement au lieu d'attendre la fin de l'impression... On peut travailler sous FORTH mais il faut quand même attendre la fin de l'impression pour en commander une autre!

IV) FICHIERS ALLOCATE.SCR ET RELOCATE.SCR

1) ALLOCATE.SCR

Les mots MALLOC et MFREE permettent de réserver ou de libérer des zones mémoire (en dehors de la zone FORTH).

MALLOC (d --- laddr) réserve d octets (d = entier 32 bits) et retourne l'adresse absolue (32 bits) du début de zone réservée.

MFREE (laddr ---) libère la zone précédemment réservée commençant à l'adresse laddr.

Pour travailler sur des zones mémoire extérieures au FORTH on dispose des mots suivants:

>ABSADDR (addr --- laddr) convertit l'adresse 16 bits addr interne au FORTH en laddr adresse absolue 32 bits.

L@ L! LC@ LC! (ndlr: implantés également sur TURBO-Forth) et L2! L2@ LCMOVE LDUMP opèrent comme leur homologue @ ! C@ C! 2@ 2! CMOVE DUMP du standard F83 mais sur des adresses absolues 32 bits.

>ABSADDR est défini dans DIVERSES.SCR, les autres dans le noyau FORTH-83.SCR.

2) RELOCATE.SCR

Les mots RELOCATE et BUFFERS permettent de remodeler la disposition des 64ko de la zone FORTH.

RELOCATE (n1 n2 ---) fixe la longueur maximale de la pile de données à n1 et celle de la pile de retour à n2.

BUFFERS (n ---) fixe à n (n>0) le nombre de blocs de 1024 octets réservés en haut de la mémoire pour le stockage des écrans. Initalement ce nombre est fixé à 10, en le diminuant, on augmente la place disponible pour le dictionnaire, au prix d'un confort moindre en mode éditeur (lectures et écritures plus fréquentes sur le disque...).

Avant l'appel de ces deux mots, il est conseillé de faire SAVE pour verrouiller le dictionnaire et les vocabulaires.

V) TRADUCTION DES MESSAGES EN ALLEMAND

HAEH? le mot qui précède HAEH? n'a pas été trouvé dans le dictionnaire (vérifier le vocabulaire de recherche!).

NEIN la liste des écrans que devait copier CONVEY est vide ou trop longue.

DATEI NICHT GEFUNDEN un fichier dont l'ouverture a été demandée (par USE par exemple...) n'a pas été trouvé sur la disquette. vérifier PATH

DISK SCHREIBGESCHUTZT essai d'écriture sur une disquette protégée (volet ouvert).

DISK VOLL disquette pleine

PFAD NICHT GEFUNDEN le chemin de recherche du fichier défini par DIR n'existe pas.

UNGULTIGE HANDLE# erreur de gestionnaire de fichier: souvent appel d'un fichier déjà fermé.

UNGULTIGE LAUFWERK le lecteur de disquette défini par SETDRIVE ou A: B: n'existe pas.

ZUGRIF NICHT MÖGLICH appel d'un écran qui n'existe pas. Exemple: 1 10 THRU (compilation des écrans 1 à 10) d'un fichier pour lequel CAPACITY vaut 10 (c.à.d. écrans 0 à 9 seulement!).

J'espère que cette introduction vous permettra de mieux utiliser le VOLKSFORTH-83. Il reste encore beaucoup à découvrir par vous-même (la programmation sous GEM par exemple...) et vous apprendrez des tas de chose en parcourant les fichiers sources dont certains sont documentés (mais hélas en allemand!).

■ Bon amusement!
Daniel FLIPO

FORTH

LA VIRGULE FLOTTANTE 83-STANDARD

V1.0 d'après version fig MPE Ldt
adaptée F83 par M.ZUPAN

pour TURBO-Forth
F83 Laxen et Perry CP/M et MSDOS
F83 ORIC

INFORMATION IMPORTANTE: ce programme est disponible en disquette auprès de l'ASSOCIATION JEDI contre envoi de la somme de 37,00 Fr (10 timbres à 3,70 Fr) et accompagné de divers autres programmes déja diffusés dans JEDI.

LE LISTING DU PROGRAMME:

```
ONLY DEFINITIONS FORTH VOCABULARY F-PACK

\ ******************************************************
\ routines <-S S-> de décalages 32 bits
\ ******************************************************
\ Ces deux routines sont au coeur de tout système
\ flottant:
\ elles opèrent les décalages à gauche et à droite
\ de 1 bit avec introduction d'un bit 0 ou 1 à un bout
\ et récupération d'un bit 0 ou 1 à l'autre
\ bout. Elles peuvent être écrites en Forth comme
\ ci-dessous mais il est hautement recommandé de les
\ écrire en code. Le reste de F-PACK sera en haut niveau.

\ : <-S  (S ud bit-entrée-D  --  ud bit-sortie-G )
\        -ROT DUP 0< >R 2DUP D+ ROT 0 D+ R> ABS ;
\ : S->  (S ud bit-entrée-G  --  ud bit-sortie-D )
\        >R 2 MU/MOD R> IF 0 32768 D+ THEN ROT ;

CODE <-S
     AX POP   AX RCR   DX POP   BX POP
     BX RCL   DX RCL   0 # AX MOV   AX RCL
     BX PUSH   2PUSH   END-CODE

CODE S->
     AX POP   AX RCR   DX POP   BX POP
     DX RCR   BX RCR   0 # AX MOV   AX RCL
     BX PUSH   2PUSH   END-CODE

\ ******************************************************
\ variables et constantes
\ ******************************************************
: F!     (S f adr -- )
         DUP >R 2! R> 4 + ! ;
: F@     (S adr -- f )
         DUP >R 4 + @ R> 2@ ;

: FVARIABLE CREATE   0 , 0 , 0 ,   DOES> ;
: FCONSTANT CREATE HERE 6 ALLOT F! DOES> F@ ;

VARIABLE EXP       VARIABLE  SGN       VARIABLE  FLG
VARIABLE EXP1      VARIABLE LO1      VARIABLE LO2
VARIABLE HI1       VARIABLE HI2      VARIABLE *SN
VARIABLE /SN       VARIABLE +SN1     VARIABLE +SN2
VARIABLE %N        VARIABLE %EXP     VARIABLE %SGN
VARIABLE %FLG      VARIABLE FIX

      0      0     0    FCONSTANT %0
      0  16384     1    FCONSTANT %1
  26214  26214    -3    FCONSTANT %.1
      0  20480     4    FCONSTANT %10
      0  23040     8    FCONSTANT %180
  -4780  25735     2    FCONSTANT PI
  -4781  25735     1    FCONSTANT PI/2
  -4780  25735     0    FCONSTANT PI/4
 -15946  20860     0    FCONSTANT 2/PI
 -15946  20860    -1    FCONSTANT 1/PI
  31129  23170     0    FCONSTANT %SQR.5

FVARIABLE FST1    FVARIABLE FST2    FVARIABLE FST3
FVARIABLE DUM     FVARIABLE %STK1    FVARIABLE %Y1
FVARIABLE %F
```

```
\ ******************************************************
\ opérations élémentaires
\ ******************************************************

: DNORM   (S d n -- mantisse exp )
\ normalise n bits d'un double
          EXP ! 2DUP OR
            IF DUP 0< SGN ! DABS
               BEGIN  EXP 1-! 0 <-S DROP
                      DUP 0<
               UNTIL
               EXP 1+! 0 S-> DROP
               SGN @
               IF DNEGATE THEN
               EXP @
             ELSE 0
             THEN ;

: D>F      (S d -- f )
           31 DNORM ;
: S>F      (S n -- f )
           S>D D>F ;
: FSIGN    (S f -- f f1 )
           >R DUP 0< >R DABS R> R> SWAP ABS ;

: FDUP     3DUP ;
: FOVER    5 PICK 5 PICK 5 PICK ;
: FROT     FST1 F! FST2 F! FST3 F!
           FST2 F@ FST1 F@ FST3 F@ ;
: FSWAP    FST1 F! FST2 F! FST1 F@ FST2 F@ ;
: FNEGATE >R DNEGATE R> ;
: FDROP    2DROP DROP ;
: FABS     >R DABS R> ;

: D*       (S d1 d2 -- d1*d2 64 bits )
           HI2 ! LO2 ! HI1 ! LO1 !
           LO1 @ LO2 @                UM*
           0    LO2 @ HI1 @           UM* D+
                LO1 @ HI2 @           UM* D+
                 0    HI1 @ HI2 @     UM* D+ ;

: D!       DUM 2! ;
: D@       DUM 2@ ;

: F*       (S f1 f2 -- f3 ) \ multiplication
           FSIGN *SN ! 1+ EXP ! D!
           FSIGN *SN +! EXP +! D@
           D* 2DUP OR
           IF BEGIN
                DUP 16384 AND 0= WHILE
                EXP 1-! >R >R 0 <-S R> R> ROT <-S DROP
              REPEAT
              D! NIP D@ ROT 0< ABS 0 D+
              DUP 0<
               IF 0 S-> DROP EXP 1+! THEN
              *SN @ 1 =
               IF DNEGATE THEN
              EXP @
            ELSE FDROP DROP 0 0 0
            THEN ;

: F/       (S f1 f2 -- f3 ) \ division
           FSIGN /SN ! NEGATE EXP !
           2DUP OR
           IF DNEGATE D! FSIGN /SN +! EXP +!
              >R >R 0 0 R> R>
              31 FLG !
              BEGIN D@ D+ DUP 0<
                IF    >R >R 0 <-S DROP R> R> D@ DNEGATE D+
                ELSE  >R >R 1 <-S DROP R> R>
                THEN
                0 <-S DROP FLG DUP 1-! @ 0=
              UNTIL
              2DROP EXP @ 1+ DNORM /SN @ 1 =
              IF FNEGATE THEN
           ELSE TRUE ABORT" division par zéro"
           THEN ;

: F+       (S f1 f2 -- f3 ) \ addition
           2 PICK 2 PICK OR
           IF FSIGN +SN1 ! EXP ! D! 2 PICK 2 PICK OR
```

```
        IF FSIGN +SN2 ! EXP1 ! EXP @ EXP1 @ - DUP
           IF DUP 0> IF 0 DO 0 S-> DROP LOOP
                     ELSE D@ ROT ABS 0
                          DO 0 S-> DROP LOOP D!
                     THEN
           ELSE DROP
           THEN
           D@ +SN1 @ +SN2 @ - EXP @ EXP1 @ MAX EXP !
           IF DNEGATE D+
           ELSE D+ EXP 1+! 0 S-> 0 D+ DUP 0<
                IF EXP 1+! 0 S-> DROP THEN
           THEN
           +SN2 @ IF DNEGATE THEN EXP @ DNORM
        ELSE FDROP D@ +SN1 @
             IF DNEGATE THEN EXP @ THEN
     ELSE FDROP
     THEN ;

: F-     (S f1 f2 -- f3 ) \ soustraction
         FNEGATE F+ ;

: F>D    (S f -- d )
         FSIGN SGN ! DUP 0<
         IF FDROP 0 0
         ELSE 31 SWAP - DUP 0<
             ABORT" dépassement pour F>D"
             0 ?DO 0 S-> DROP LOOP
             SGN @ IF DNEGATE THEN
         THEN ;

: FINT   (S f -- f )
         DUP 32 < IF F>D 31 DNORM THEN ;

: F>S    (S f -- n )
         F>D ?DUP IF 1+
                 ABORT" dépassement pour F>S"
              THEN ;

: F0<    DROP NIP 0< ;
: F0>    DROP NIP 0> ;
: F0=    DROP OR  0= ;
: F=     F- F0= ;
: F<     F- F0< ;
: F>     F- F0> ;
: FMAX   FOVER FOVER F< IF FSWAP THEN FDROP ;
: FMIN   FOVER FOVER F> IF FSWAP THEN FDROP ;

: FLITERAL          \ compilation d'un flottant
         STATE @
         IF ROT [COMPILE] LITERAL SWAP
                [COMPILE] LITERAL
                [COMPILE] LITERAL
         THEN ; IMMEDIATE

: F/MOD  (S f1 f2 -- rf3 qf4 )
\ division décimale et reste
         FOVER FOVER F/ FROT FROT FABS FSWAP
         FSIGN >R FSWAP
         8 PICK 8 PICK 8 PICK FABS FINT F* F- R>
         IF FNEGATE THEN FSWAP ;
: FMOD   F/MOD FDROP ;

\ **********************************************************
\ entrées/sorties : conversions alphanumériques-flottants
\ **********************************************************

\ entrées : E ... FLOAT , VAL , F'
: E
\ conversion mantisse d'un nombre simple ou double
         DPL @ -1 = IF S>D 0 ELSE DPL @ NEGATE THEN EXP1
! D>F ;
: FLOAT  \ incorporation de l'exposant
         EXP1 @ + DUP
         IF DUP 0> IF     0 DO %10 F* LOOP
                   ELSE ABS 0 DO %.1 F* LOOP THEN
         ELSE DROP THEN ;
: VAL    (S string -- f )
\ conversion d'une chaîne en flottant
         BASE @ >R DECIMAL
         ASCII + SKIP
         2DUP ASCII E SCAN >R >R
         DROP 1- (NUMBER?) DROP
         DPL @ -1 = IF DROP THEN E
         R> R> ?DUP IF ASCII E SKIP ASCII + SKIP
                       DROP 1- (NUMBER?) 2DROP
                    ELSE DROP 0
                    THEN
         FLOAT R> BASE ! ;

: F'     (S <chaîne> -- f )
\ interprète le mot suivant comme flottant
         BL   WORD   COUNT   VAL  [COMPILE]  FLITERAL  ;
IMMEDIATE

\ sorties : (E.) (F.) E. F.
:  ROUND       \  arrondit  à  zéro  les  bits  les  moins
significatifs
         FSIGN >R >R SWAP DUP 224 AND 0=
           IF -256 AND THEN
         DUP 224 AND 224 - 0=
           IF -256 AND 256 + DUP 0=
             IF SWAP 1+ ELSE SWAP THEN
           ELSE SWAP THEN
         R> OVER 0< IF SWAP 0 2 UM/MOD NIP SWAP 1+ THEN
         R> IF FNEGATE THEN ;
: ?10PWR \ extraction des puissances de dix
         DUP 0<
         IF    DUP DUP 10 / DUP 3 * >R 10 * - 3 / R> +
         ELSE DUP DUP 10 / DUP 3 * >R 10 * - DUP 7 /
              DUP >R 7 * - 4 / R> + R> +
         THEN ;
: OPREP  \ conversion double décimal
         3 ?ENOUGH FSIGN %SGN !
         ?10PWR DUP %EXP ! 8 SWAP - DUP 0=
         IF DROP
         ELSE DUP 0<
            IF ABS 0 DO %.1 F* LOOP
            ELSE   0 DO %10 F* LOOP
            THEN
         THEN
         DUP 29 > IF %.1 F* %EXP 1+! THEN
         ROUND  F>D ;
: #F     \ digit décimal, zéros significatifs
         10 MU/MOD ROT
         DUP 0<> %FLG @ 0= OR
         IF ASCII 0 + HOLD %FLG OFF ELSE DROP THEN ;

: (E.)   (S f -- string ) \ chaîne notation scientifique
         OPREP
         <# %FLG OFF %EXP @ DUP ABS 0 #F #F 2DROP
            0< IF ASCII - ELSE ASCII + THEN HOLD
            ASCII E HOLD %FLG ON
            8 0 DO #F LOOP ASCII . HOLD %FLG OFF #F
            %SGN  @ 0<> SIGN
         #> ;

: (F.)   (S f -- string )
\ chaîne notation décimale libre ou fixée
         DUP -10 27 BETWEEN
         IF  OPREP
            <# 8 %EXP @ - 0 MAX
               FIX @ ?DUP
               IF 0 MAX 8 MIN 2DUP MIN >R - DUP
                     0< IF ABS 0  DO ASCII 0 HOLD LOOP
                        ELSE   0 ?DO 10 MU/MOD ROT 5 >
                                     IF 1 0 D+ THEN
                               LOOP
                        THEN R> %FLG OFF
               ELSE  %FLG ON
               THEN
               0 ?DO #F LOOP
               %FLG @ IF %FLG OFF ELSE ASCII . HOLD THEN
               BEGIN #F 2DUP OR 0= UNTIL
               %SGN @ 0<> SIGN
            #>
         ELSE (E.) THEN ;

: E.     (E.) TYPE SPACE ;
: F.     (F.) TYPE SPACE ;

: E.R    >R (E.) R> OVER - SPACES TYPE ;
: F.R    >R (F.) R> OVER - SPACES TYPE ;

\ **********************************************************
\ fonctions transcendantes
\ **********************************************************
```

```
: FSQR     3 ?ENOUGH 2 PICK 2 PICK OR
           IF OVER 0<
              ABORT" racine carrée d'un nombre négatif"
              0 SWAP %N ! FDUP %F F!
              23781  19338  0 F*
              -11264 27348 -1 F+ %Y1 F!
              2 0 DO %F F@ %Y1 F@ F/ %Y1 F@ F+
                   1- %Y1 F! LOOP
              %N @ 1 AND IF %N 1+! %Y1 F@ %SQR.5 F*
                         ELSE %Y1 F@
                         THEN
              %N @ 2/ +
           THEN ;
: F1/X     %1 FSWAP F/ ;
: DEG>RAD %180 F/ PI F* ;
: RAD>DEG %180 F* PI F/ ;
: (SIN)    FDUP FDUP F* FDUP
           21489   26278 -25 F*
           -22679  23644 -18 F+ FOVER F*
           -340   -26631 -12 F+ FOVER F*
           17224   17476  -6 F+ FOVER F*
           -21843 -21846  -2 F+ F* %1 F+ F* ;
: SINFN    FDUP -8192 25735 18 F>
           ABORT" dépassement trigonométrique"
           FDUP 1/PI F*
           FINT FDUP F>S %N !
           PI F* F-
           FDUP PI/2 F>
             IF PI F- %N 1+! THEN
           (SIN) %SGN @ %N @ 1 AND IF NEGATE THEN
           0< IF FNEGATE THEN ;
: FSIN     DEG>RAD FSIGN IF -1 ELSE 1 THEN %SGN !
           SINFN ;
: FCOS     DEG>RAD PI/2 F+ FSIGN IF -1 ELSE 1 THEN %SGN !
           SINFN ;
: (TAN)    FDUP FDUP F* FDUP
           12781  19958  -6  F*
           24066  24331  -8  F+ FOVER F*
           23774  25758  -5  F+ FOVER F*
           24849  27983  -4  F+ FOVER F*
            567   17484  -2  F+ FOVER F*
           13558  21845  -1  F+ F* %1 F+ F* ;
: FTAN     DEG>RAD FSIGN %SGN ! FDUP
           -8192 25735 18 F> ABORT" dépassement tangente"
           FDUP 2/PI F* FINT FSWAP FOVER PI/2 F* F-
           FDUP PI/4 F> IF PI/2 F- (TAN) FSWAP %1 F+
                        ELSE (TAN) FSWAP
                        THEN
           F>S 1 AND IF F1/X FNEGATE THEN
           %SGN @ IF FNEGATE THEN ;
: ARCFN    FDUP
           -5628  -21182  -9 F*
           25269   27976  -7 F+ FOVER F*
           -13001 -17919  -5 F+ FOVER F*
           31784   32392  -5 F+ FOVER F*
           14038  -26306  -4 F+ FOVER F*
           20163   23325  -3 F+ FOVER F*
           6939   -28128  -2 F+ FOVER F*
           -4808   25735   1 F+ FSWAP %1
           FSWAP F- FSQR F* ;
: ATFN     FDUP FDUP F* FDUP
           -23342  24043  -8 F*
           -231   -16952  -5 F+ FOVER F*
           -289    22496  -4 F+ FOVER F*
           17866  -19737  -3 F+ FOVER F*
           -15957  27934  -3 F+ FOVER F*
           7296   -18624  -2 F+ FOVER F*
           -3476   26205  -2 F+ FOVER F*
           -13768 -21846  -1 F+ F* %1 F+ F* ;
: FASIN    3 ?ENOUGH FSIGN %SGN ! ARCFN PI/2 FSWAP F-
           %SGN @ IF FNEGATE THEN
           RAD>DEG ;
: FACOS    3 ?ENOUGH FSIGN %SGN ! ARCFN
           %SGN @ IF PI FSWAP F- THEN
           RAD>DEG ;
: FATAN    3 ?ENOUGH FSIGN %SGN ! FDUP %1 F>
           IF F1/X ATFN PI/2 FSWAP F-
           ELSE ATFN THEN
           %SGN @ IF FNEGATE THEN
           RAD>DEG ;
: FLN      3 ?ENOUGH 2 PICK 2 PICK OR 0= 2 PICK 0< OR
           ABORT" logarithme impossible"
           1 SWAP 1- %N ! %1 F- FDUP
           -8219  -27069  -7 F*
           -15445 18920   -4 F+ FOVER F*
           -1809 -24991   -3 F+ FOVER F*
           -16097 21974   -2 F+ FOVER F*
           -30256 -31554  -2 F+ FOVER F*
           -14355 21744   -1 F+ FOVER F*
           16346 -32760   -1 F+ FOVER F*
           -7680  32767    0 F+ F* %N @ S>F
           3066   22713    0 F* F+ ;
: FLOG     FLN -5035 28461 -1 F* ;
: 10^X     FDUP  ( primitive : ne pas utiliser )
           19334  31294 -10 F*
           13450   21432 -8 F+ FOVER F*
           24130   18267 -5 F+ FOVER F*
           -9172   19123 -3 F+ FOVER F*
           -4108   16671 -1 F+ FOVER F*
           23960   21716  0 F+ FOVER F*
           -14364  18862  1 F+ F*
           %1 F+ FDUP F* ;
: E^X      FDUP  ( primitive : ne pas utiliser )
           -8250 -18968 -12 F*
           -18510 22311  -9 F+ FOVER F*
           -13978 -17410 -6 F+ FOVER F*
           29320  21840  -4 F+ FOVER F*
           -10120 -21846 -2 F+ FOVER F*
           -342    32767 -1 F+ FOVER F*
           2      -32768  0 F+ F*
           %1 F+ F1/X ;
: FALN     3 ?ENOUGH FSIGN %SGN ! FDUP
           0 25600 8 F> ABORT" dépassement ALN"
           FDUP 7572 23637 1 F* FDUP F>S %N !
           FINT 3068 22713 0 F* F-
           E^X %N @ + %SGN @ IF F1/X THEN ;
: FALOG    3 ?ENOUGH FSIGN %SGN ! FDUP
           0 25344 7 F> ABORT" dépassement ALOG"
           FDUP 15402 27213  2 F* FDUP F>S %N !
           FINT 19775 19728 -1 F* F-
           10^X %N @ + %SGN @ IF F1/X THEN ;
: FX^N     4 ?ENOUGH ?DUP
           IF DUP 0< %FLG ! ABS >R %1 FSWAP R>
              BEGIN DUP 0> WHILE
                DUP 2/ DUP 2* ROT -
                IF >R FSWAP FOVER F* FSWAP R> THEN
                >R FDUP F* R>
              REPEAT
              DROP FDROP %FLG @ IF F1/X THEN
           ELSE FDROP %1
           THEN ;
: FX^Y     6 ?ENOUGH OVER 3 PICK OR
           IF OVER 0< >R FABS
              FDUP FDUP FINT F- %STK1 F!
              F>S >R FDUP R> FX^N FSWAP
              %STK1 F@ OVER 3 PICK OR
              IF 4 PICK 0< ABORT" X^Y complexe"
                 FSWAP FLOG F* FALOG F*
              ELSE FDROP FDROP
              THEN R> IF F1/X THEN
           ELSE FDROP FDROP %1
           THEN ;

EOF
```

V O C A B U L A I R E   F - P A C K
V I R G U L E   F L O T T A N T E

F-PACK est un module F83 de gestion des nombres en virgule flottante adapté du 'Forth portable floating point package' du domaine public proposé en fig par MicroProcessor Engineering Ltd Southampton.

L'écriture en est entièrement en Forth F83 pour permettre une portabilité facile et pour autoriser sa réécriture dans une précision supérieure en suivant les mêmes algorithmes.

Pour autant que la rapidité puisse être un critère quand on choisit le calcul en virgule flottante, certains mots pourront être redéfinis en code. Vous constaterez toutefois que le Forth est bien un langage rapide puisque les performances de ce module sont très honorables sans code machine ni coprocesseur.

Les nombres flottants sont codés sur 48 bits directement exploitables sur la pile paramètres du Forth. Ils occupent donc 3 cellules de la pile à la manière d'un calcul en "triple précision". On trouve ainsi toute une série de mots tels que FDUP FDROP FOVER FROT qui manipulent la pile par ensembles de 3 cellules.

Un nombre flottant est défini par son signe sur 1 bit, sa mantisse sur 31 bits, ces deux derniers formant les 32 premiers bits soit deux cellules, et enfin son exposant en puissance binaire entière signée dans la dernière cellule de 16 bits. On utilise la technique de la mantisse normalisée à gauche qui consiste à cadrer tous les bits significatifs comme derrière une virgule imaginaire en précisant dans l'exposant le rang de la puissance 2 de la partie entière.

Ainsi le nombre 10 s'écrit 1010 en binaire ce qui donnera comme flottant :

```
signe      0
mantisse  .1010000000000000000000000000000 ( 31 bits )
exposant     ^ = 4
```

le nombre flottant 10 s'écrira sur la pile

```
          0000000000000000
          0101000000000000
          0000000000000100

c.a.d :      0          20480          4
```

Il n'est bien sûr pas nécessaire de comprendre le complexe codage des nombres flottants ainsi que les opérations requises dans les calculs pour manipuler ces flottants aussi facilement que des entiers simple précision sur la pile.

Ce type de codage autorise une précision de neuf chiffres significatifs avec une gamme d'exposants en base 10 de -99 à +99.

Il existe un grand nombre de façons de déposer un flottant sur la pile : on a cherché à rester le plus simple possible au regard de la syntaxe Forth.

1) à partir d'un entier simple précision 235 S>F donne le flottant 235 ou 2.35E+2.

2) à partir d'un entier double précision 235. D>F donne également le flottant 235 notez toutefois que 2.35 donne toujours 235 car le point n'est pas significatif dans un nombre double précision.

3) avec la séquence 'nombre E exposant FLOAT' 2.35 E 0 FLOAT donne le flottant 2.35 235 E -2 FLOAT donne aussi 2.35. Notez que cette séquence prend en compte la position du point décimal dans le double précision avant E.

E agit également sur un simple précision ce qui rend le point facultatif à condition de ne pas entrer un nombre dépassant +/- 32768 limite des entiers simple précision. E est obligatoire avant FLOAT et les quatre éléments de cette syntaxe doivent évidemment être séparés par des espaces.

Cette interprétation post-fixée n'est pas valide pour la compilation des flottants.

```
4) avec F' F' 2.35  donne 2.35
        F' 2E2   donne 200.
```

Cette interprétation pré-fixée est la plus pratique. F' interprète le mot qui le suit comme un flottant. Ce mot ne doit pas contenir d'espaces mais peut contenir des indications de signes. Le point décimal est facultatif dans la mantisse avec la même restriction que pour E.
F' est utilisable pour la compilation d'un flottant.

Les écritures suivantes sont valables :

```
F' 34.03E-2
F' -140E+2
F' +34.34567
F' +2E+20
F' 3
F' 345678.
```

5) à partir d'une chaîne explicite " 34.56E-2" VAL donne le flottant 0.3456.

Les règles sont les mêmes que pour F' notamment en ce qui concerne l'absence d'espace dans la chaîne numérique.

Pour l'affichage des flottants, E. affiche le flottant sur la pile en notation scientifique c'est à dire avec un seul chiffre avant la virgule et la puissance de dix :

```
F' 345  E.  affiche  3.45E+02
```

alors que F. l'affiche en notation décimale usuelle :

```
F' 34.5 F.  affiche  34.5
```

La variable FIX permet en outre d'utiliser F. pour un affichage en virgule fixe, pratique pour aligner la virgule. Quand FIX est OFF (égal à zéro) l'affichage est libre avec toutes les décimales significatives. Si FIX est à n, le nombre affiché comporte obligatoirement n décimales:

```
FIX OFF   F' 2      F.  affiche 2
          F' 2.347  F.  affiche 2.347
5 FIX !   F' 2      F.  affiche 2.00000
          F' 2.347  F.  affiche 2.34700
2 FIX !   F' 2      F.  affiche 2.00
          F' 2.347  F.  affiche 2.35
```

Si F. ne peut afficher un nombre trop petit ou trop grand, c'est l'affichage scientifique qui est donné.
Enfin E.R et F.R rappellent .R ou D.R pour un affichage cadré à droite:

```
F' 2.35 10 F.R affiche '      2.35'
F' 2.35 10 E.R affiche '  2.35E+00'
```

On utilisera les primitives (E.) et (F.) pour traiter des flottants comme des chaînes, exactement comme (.) ou (D.) en F83 standard.

Le vocabulaire F-PACK contient un ensemble homogène facilement compréhensible de fonctions et opérateurs sur les nombres flottants:

1) manipulation de la pile

```
       f    FDUP    f  f
    f1 f2   FOVER   f1 f2 f1
 f1 f2 f3   FROT    f2 f3 f1
    f1 f2   FSWAP   f2 f1
       f    FDROP
```

2) conversions

```
    n    S>F    f
    d    D>F    f
    f    F>S    n
    f    F>D    d
```

3) tests

```
       f    F0=    f1
       f    F0>    f1
       f    F0<    f1
    f1 f2   F=     f1
    f1 f2   F>     f1
    f1 f2   F<     f1
```

4) opérateurs dyadiques

```
    f1 f2   F+      f=f1+f2
    f1 f2   F-      f=f1-f2
    f1 f2   F*      f=f1*f2
    f1 f2   F/      f=f1/f2
    f1 f2   F/MOD   f3=f1-f2*int(f1/f2)  f4=f1/f2
    f1 f2   FMOD    f3 ci dessus
```

f1 f2	FX^Y	f=f1^f2
f1 n	FX^N	f2=f1^n
f1 f2	FMIN	f=min(f1,f2)
f1 f2	FMAX	f=max(f1,f2)

5) opérateurs monadiques

f	FNEGATE	f'=-f
f	FINT	f'=int(f)
f	FABS	f'=abs(f)
f	FSQR	f'=sqr(f)
f	F1/X	f'=1/f
f°	DEG>RAD	frad
frad	RAD>DEG	f°
f	FSIN	f'=sin f
f	FCOS	f'=cos f
f	FTAN	f'=tan f
f	FASIN	f'=arcsin f
f	FACOS	f'=arcos f
f	FATAN	f'=arctan f
f	FLN	f'=Ln(f)
f	FLOG	f'=log(f)
f	FALN	f'=e^f
f	FALOG	f'=10^f

6) compilation et divers forth

f	FLITERAL	en compilation : compile f
f	FCONSTANT	<mot> , à exécution de mot : --f
	FVARIABLE	<mot> , à exécution de mot : --adr
f adr	F!	
adr	F@	f

7) constantes et variables

f-constantes	%0 %1 %.1 %10 PI PI/2 %E
f-variables	FST1 FST2 FST3 utilisables en accumulateurs temporaires

Nous ne voudrions pas terminer cette présentation de F-PACK sans vous rappeler la devise: FLUCTUAT NEC MERGITUR. Les flottants c'est bien mais ne vous laissez pas submerger. Forth est beaucoup plus à l'aise avec des entiers et voyez toujours si votre application ne serait pas plus puissante avec une écriture en entiers ou en virgule fixe: vous y gagnerez notamment en rapidité. ■

COURRIER: TURBO-FORTH, UN STANDARD?

En tant que co-auteur de TURBO-FORTH, j'aimerais faire part ici de quelques réflexions d'ordre "philosophique":

FORTH n'est probablement pas un langage, encore moins un standard: c'est plutôt un état d'esprit, un concept. BASIC est un langage qui eut son ou ses standards: on a fait des merveilles avec des GOSUB 13440. Et puis on s'est aperçu que la programmation structurée faisait gagner du temps au programmeur. Où est maintenant le standard BASIC? Un standard ne vit que le temps où il est reconnu comme standard, le temps qu'un autre standard prenne sa place.

L'esprit FORTH est tout autre: s'adapter d'une façon naturelle (mais oui, la logique polonaise d'une pile est naturelle: comment croyez-vous que font tous les autres langages "dedans"?), s'adapter donc à tout ce qui fait une machine dans un seul but: développer au plus près, vite et bien.

Un ordinateur est un ensemble d'adresses, un ou plusieurs processeurs, un environnement, un système d'exploitation et des programmes. Vouloir en faire abstraction relève de la science théorique: la machine de TURING est certainement capable elle aussi de calculer les transformées de Fourier...

Il n'est plus possible aujourd'hui de considérer un langage informatique comme un quelconque Esperanto de la programmation. Avec TURBO-FORTH MS-DOS, nous sommes partis plus modestement d'une machine donnée, d'un système d'exploitation qui est ce qu'il est, et d'une couche logicielle formant un ensemble qui LUI s'est imposé comme un standard de fait.

Nous avons abandonné le concept Forth des blocs: il partait du principe pré-CP/M que Forth devait posséder son propre système d'exploitation des secteurs du disque. Pour gérer quoi? Mieux qu'un véritable DOS? Soyons sérieux et cessons de nous enfermer dans une tour d'ivoire où il nous faut sans cesse ré-inventer la roue. TURBO-FORTH fait allégeance au standard MS-DOS et à tout son environnement: c'est ça la vraie magie de l'esprit Forth.

Nous avons abandonné l'éditeur interne: venez me voir programmer mon TURBO-FORTH configuré WordPerfect avec ses macros et je ne vous donne pas cinq minutes pour vous exclamer "révolutionnaire!". Et croyez bien que je n'étais pas peu fier de mon éditeur plein-écran de blocs! Seulement voilà, WP est WP: combien de temps me faudrait-il pour faire le dixième de ce qu'il fait? Je n'ai pas l'impression de piloter un CRAY-1, mon cher Jaccomard, je ne cherche qu'à travailler vite et bien. Les 15 jours d'apprentissage de WP sont amortis depuis longtemps!

Mais nous irons encore plus loin dans l'ouverture.

Un jour que je m'appliquais à écrire pour Turbo la gestion des fichiers en accès direct avec champs et tout, me voyant déjà l'heureux concepteur de FBASE-III, mon collègue PETREMANN qui s'y connaît eut cette remarque sans appel: "dBASE III/III+(/IV?) est IN-CON-TOUR-NA-BLE!". Le standard c'est donc lui: ou Turbo importe, traite et exporte l'environnement dBASE, ou on en reste sagement à l'excellent FBASE-II. On ne peut être plus clair.

Quand au standard Forth lui-même, il n'est pas question pour l'instant de renier F83. Nous avons constamment cherché à respecter l'esprit et la lettre de F83 dans TURBO-FORTH, au prix même parfois de quelques concessions à l'ancêtre. Et si nous avons ajouté de nouveaux concepts, ils restent droits dans la ligne du standard. Ainsi, le concept du $EXECUTE réentrant achève la mise à mort d'une dichotomie surannée entre interprétation et compilation que Forth a toujours instinctivement refusée.

Nous comptons bien que dans cinq ans TURBO-FORTH MS-DOS soit totalement dépassé face aux nouveaux concepts qui ne vont pas manquer d'apparaître. Probablement sommes-nous déjà à la traîne. Mais nous avons l'outil et l'esprit: si l'avenir est par là, donnez-nous un 386, un mois tous frais payés aux Seychelles, un bon manuel sur ce qu'il y a dans la bécane et son DOS et nous vous métacompilerons un "FULL-SPEED-FORTH-OS/2-Windows-machin" en bien moins de temps qu'il n'en faut à Borland pour sortir un TURBO-C bogué.

Et croyez-bien qu'il ne s'agirait absolument pas d'un Forth "tournant" sous OS/2: ça n'aurait aucun intérêt. Ou alors à quoi bon acheter un tel système? Ce Forth n'aurait qu'une partielle ressemblance avec l'ancien: les principaux mots étant bien évidemment les plus spécifiques du nouveau ...standard!

Franchement le F83 est génial... sur mon ATMOS (normal: c'est moi qui l'ai métacompilé!). Mais d'emblée j'ai refusé pour mon PC un standard qui tourne tout aussi bien sur mon vieux 8-bits. Logique non? Comprenez-moi: je ne dis pas qu'il faille sans cesse sacrifier aux derniers cris à la mode (salut amical à tous les oriciens qui ont si bien alimenté le numéro 40 de JEDI et prouvent ainsi que la valeur n'attend pas le nombre des kilo-octets),

suite page 12

Objectif: Calcul en quadruple précision.
Langage: Assembleur 6502 et F83
Matériel ATMOS mais...
date: 15/1/88

Quadruple précision

Si Forth ne connait pas pour le F83 les nombres flottants,il sait faire des opérations en double précision. Cela peut suffire dans bien des cas mais pas toujours. Le but de ces quelques écrans est de pousser la balle un peu plus loin en lui apprenant la quadruple précision. Il s'agit donc de faire des calculs sur 4 mots de 16 bits .
Attention,tous les calculs ne seront pas possibles,on garde la logique:jusqu'à présent D* (32bits * 32 bits) n'existait pas;il va exister maintenant mais pas Q* (64 * 64 bits) car le résultat pourrait aller jusqu'à 128 bits.
Logique tout ça !
Au fait combien ça fait 424935072 * 820345021 . Prenez donc votre calculatrice !
Commençons par la fin ! Ecrans 8 et 9
Ils contiennnent le nécessaire pour que l'interprêteur puisse reconnaître des nombres quadruples.
Jusque là il sait reconnaître des nombres doubles et tous les calculs effectués par NUMBER sont doubles. Il faut donc ré écrire ces mots.
A ce propos, tout le monde sait rentrer directement un nombre double par 23. ou .23 ou 2.3 par exemple. Il faut donc aussi un caractère de reconnaissance pour les quadruples. Vous savez tous (mais pas moi) qu'en fait F83 a 4 symboles pour les doubles (,.-/) ;maintenant il n'en aura que trois (,.-) car j'ai réservé le symbole (/) pour ces nouveaux nombres. 123/ met sur la pile 4 mots de 16 bits 123 0 0 0
Ce symbole / est prioritaire dans le sens que 123/456. est un nombre 64 bits comme 123.456/ mais 123.456 reste double.
Les mots CONVERT2 (NUMBER?2) et NUMBER?2 sont donc les nouveaux mots qui permettent de rentrer ces nombres.Tous les calculs internes sont sur 64 bits.
Ensuite ça se corse. Comment tout faire sans en faire de trop ! Normalement à la sortie de NUMBER les mots sont doubles. Le mot INTERPRET enleve la partie haute si le nombre est un simple précision ; le mot] compile de même en posant à chaque fois la question DOUBLE?.Ces mots ne sont pas vectorisés d'où difficulté.
La solution que j'ai prise fait la moitié du travail en sortie de NUMBER. Et vive la vectoristion.
La variable DPL2 prend le caractère de reconnaissance ce qui permet de le tester après. NUMBER teste alors ce caractère et 'droppe' la partie 32 bits haute si DPL2 ne contient pas '/'. Si c'est un nombre 64 bits pas de problème,il reste sur la pile mais second problème : en compilation. Il aurait fallu un mot QLITTERAL permettant de compiler ce nombre,branché dans le mot]. Je teste le cas toujours dans NUMBER et j'en compile alors la moitié;l'autre moitié le sera par].
Une définition du style : ESS 1234567890/123 987654/124 Q+ Q. ; est alors parfaitement valide.
Il y a cependant autre chose qui ne me plait guère mais j'en renvoie la faute à d'autres:Un mot 64 bits va occuper 16 octets en compilation car il est compilé comme 4 mots 16 bits précédés chacun de LIT qui permet après de le retourner sur la pile (Chez nous on a aussi CLIT on gagne 1 octet). Il aurait été plus interessant d'avoir un mot QLIT permettant de compiler tout en une fois,mais puisque DLIT n'existe pas non plus. Dans le cas de QLIT on gagnerait à chaque fois 6 octets et 2 pour DLIT.
Note Evidemment 2/ n'est pas un nombre 64 bits puisque ce mot existe .Le problème existait avant! Faire 2// ou 02/ etc.
Les primitives de base en assembleur
J'ai voulu en faire le moins possible tout en gardant une certaine rapidité,mais tout serait possible et certains mots y gagneraient comme QU/MOD et les comparaisons. Mais enfin: On travaille en forth !
Elles sont:

UQ/MOD (uq dividende, ud diviseur ------ uq quotient ,ud reste)
ressemble à UM/MOD bien qu'il ne sorte pas la même chose comparativement. Il s'agit donc de la division non signée. Attention le reste sort en premier.
La difficulté est que le 6502 ne travaille que sur des octets et qu'il y en a 8 au total. Ca fait beaucoup mais j'ai réussi à ne pas trop gaspiller de mémoire. J'utilise la zone de travail interne de A5 à AB notée N-1,N...
UQ* (ud multiplicande,ud multiplicateur ----- uq produit) ressemble à UM*.
4DROP (q ----) ou (n1,n2,n3,n4 ----)
Q+ (q1 , q2 --- q1+q2) ressemble à D+ .
QNEGATE (q1 --- -q1) ressemble à DNEGATE

Les autres opérations Ecran 5
*Q (d1, d2 --- q) est la multiplication signée qui correspond à *D
Cette opération suit la règle des signes normalement.
Elle m'a quand même fait mettre le doigt sur un problème: le mot PICK. Et je pose une question sur notre bible à tous: le livre FORTH-83 STANDARD écrit par l'excellent M.PETREMANN,le non moins excellent J-M PREMESNIL et le génial (qui travaillait sur ATMOS) M.ZUPAN.Je lis p 95 entre autres *1 PICK est équivalent à DUP* .Je cherche dans le source LAXEN et PERRY dans C:KERNEL86.BLK écran 114 *For example,if the stack has 1 2 3 then 0 PICK is 3*
Alors !!!!!

QU/MOD (q1,d1 --- q2,d2) est la division signée

Le quotient 64 bits suit la règle des signes et le reste est toujours du même signe que le diviseur et comme dit l'autre: *Division is floored* Ca y'en a vouloir dire que ça fait comme ça:

22/ 7. QU/MOD D. Q. 1 3   22/ -7. QU/MOD D. Q. -6 -4

-22/ 7. QU/MOD D. Q. 6 -4   -22/ -7. QU/MOD D. Q. -1 3

C'était un tantinet délicat à écrire. Il y a un monde fou sur les piles:ça rentre,ça sort et ça marche.

Q/ et Q/MOD sont les divisions signées qui délivrent le premier le quotient (q64) l'autre le reste (d32)

Q- fait la soustraction.

<u>L' affichage</u> écran 4

Il suit tout à fait la logique habituelle qui fait mettre sous le PAD les chiffres à afficher.Comme le maximum peut être d'une vingtaine de chiffres,il n'y a pas de problème.

UQ. affiche un nombre non signé : Essayez -1/ UQ. Ouah ça jette !

Q. affiche le même mais signé : Essayez -1/ Q. Tiens ça fait -1 !

Q.R affiche un nombre signé dans un champ justifié à droite. -1/ 20 Q.R

On pourrait faire UQ.R mais ne figure pas ici:>R (UQ.) R>OVER - SPACES TYPE

QABS donne la valeur absolue.

D>Q permet de passer un double signé en quadruple signé: -1. D>Q est équivalent à -1/

S>Q permet de passer un simple signé en quadruple signé -1 S>Q est équivalent à -1/

<u>Les comparaisons double et simple précision</u> ecran 6 et 7

Ici figurent aussi celles en double précision qui existent déjà en standard F83 mais ne figurent pas pour mon ATMOS.

D'abord les mots de traficotage de pile: 2ROT 4SWAP 4OVER 8DUP sans mystère.

Q0= (q --- f) compare le 64 bits à zéro et sort un flag en conséquence

Q= (q1,q2 -- f) compare deux nombres

Q<> (q1,q2 -- f) prend le contraire de Q=

QU< (q1,q2 -- f) compare deux 64 bits non signés entre eux

Q< et Q> (q1,q2 -- f) compare deux 64 bits signés

QMIN et QMAX retourne le minimum ou le maximum de deux 64 bits signés

4VARIABLE crée une variable capable de prendre un nombre 64 bits grace aux mots 4! et 4@

Q+! permet d'incrémenter une variable 64 bits directement.

ex: 4VARIABLE ESSAI 12/ ESSAI 4! 36/ ESSAI Q+! ESSAI 4@ Q. affiche 48

Attention le nombre à incrémenter doit être un 64 bits -> <u>36/</u>

Voilà.Bon maintenant on attaque la précision 8 puis 16... Je vais commencer à me trouver à l'étroit !

Amicalement Jean-Luc SIRET

```
A-QUADRUPLE.FTH / SCRN# 1
\ Division 64/32-> 64q 32r
ONLY FORTH DEFINITIONS ALSO HEX
CODE UQ/MOD  2 ,X LDA, A6 STA, 2 ,X STY, 3 ,X LDA, A7 STA,
             3 ,X STY, 0 ,X LDA, A8 STA, 0 ,X STY,
             1 ,X LDA, A9 STA, 1 ,X STY, A5 STY,
             40 # LDY, BEGIN,
             A ,X ASL, B ,X ROL, 8 ,X ROL, 9 ,X ROL,
             6 ,X ROL,  7 ,X ROL, 4 ,X ROL, 5 ,X ROL,
             2 ,X ROL,  3 ,X ROL, 0 ,X ROL, 1 ,X ROL, A5 ROL,
        SEC, 2 ,X LDA, A6 SBC, PHA, 3 ,X LDA, A7 SBC, PHA,
             0 ,X LDA, A8 SBC, PHA, 1 ,X LDA, A9 SBC, PHA,
             A5 LDA, 0 # SBC, CS IF,
             PLA, 1 ,X STA, PLA, 0 ,X STA, PLA, 3 ,X STA,
             PLA, 2 ,X STA, A ,X INC, CLC, CS IF, SWAP
             THEN, PLA, PLA, PLA, PLA, THEN, DEY, 0= UNTIL,
             NEXT JMP,            ;C                    -->
```

```
A-QUADRUPLE.FTH / SCRN# 2
\ UQ*   32*32-> 64
CODE UQ* 6 ,X LDA, A6 STA, 6 ,X STY,
      7 ,X LDA, A7 STA, 7 ,X STY,
      4 ,X LDA, A8 STA, 4 ,X STY,
      5 ,X LDA, A9 STA, 5 ,X STY,
      20 # LDY, BEGIN,
      6 ,X ASL, 7 ,X ROL, 4 ,X ROL, 5 ,X ROL,
      2 ,X ROL, 3 ,X ROL, 0 ,X ROL, 1 ,X ROL, CS IF, CLC,
      A6 LDA, 6 ,X ADC, 6 ,X STA, A7 LDA, 7 ,X ADC, 7 ,X STA,
      A8 LDA, 4 ,X ADC, 4 ,X STA, A9 LDA, 5 ,X ADC, 5 ,X STA,
      0 # LDA, 2 ,X ADC, 2 ,X STA,
      THEN, DEY, 0= UNTIL, NEXT JMP,            ;C

CODE 4DROP 8 # LDY, BEGIN, INX, DEY, 0= UNTIL,
      NEXT JMP,            ;C
-->
```

```
A-QUADRUPLE.FTH / SCRN# 3
\ Q+ Q- QNEGATE
CODE Q+ CLC, 6 ,X LDA, E ,X ADC, E ,X STA,
    7 ,X LDA, F ,X ADC, F ,X STA, 4 ,X LDA, C ,X ADC, C ,X STA,
    5 ,X LDA, D ,X ADC, D ,X STA, 2 ,X LDA, A ,X ADC, A ,X STA,
    3 ,X LDA, B ,X ADC, B ,X STA, 0 ,X LDA, 8 ,X ADC, 8 ,X STA,
    1 ,X LDA, 9 ,X ADC, 9 ,X STA, ' 4DROP 2+  JMP,      ;C
CODE QNEGATE SEC, TYA, 6 ,X SBC, 6 ,X STA,
    TYA, 7 ,X SBC, 7 ,X STA, TYA, 4 ,X SBC, 4 ,X STA,
    TYA, 5 ,X SBC, 5 ,X STA, TYA, 2 ,X SBC, 2 ,X STA,
    TYA, 3 ,X SBC, 3 ,X STA, TYA, 0 ,X SBC, 0 ,X STA,
    TYA, 1 ,X SBC, 1 ,X STA, NEXT JMP,                ;C


DECIMAL

-->
```

```
A-QUADRUPLE.FTH / SCRN# 4
\ #Q <#Q Q#> #QS (UQ.) UQ. 2ROT
: ?QNEGATE 0< IF QNEGATE THEN ;
: QABS DUP ?QNEGATE ;
: D>Q DUP 0< DUP ;
: S>Q DUP 0< DUP DUP ;
: 2ROT >R >R 2SWAP R> R> 2SWAP ;
: #Q BASE @ 0 UQ/MOD DROP 9 OVER < IF 7 + THEN 48 + HOLD ;
: <#Q <# ;
: Q#> 2DROP 2DROP HLD @ PAD OVER - ;
: #QS BEGIN #Q 2DUP OR >R 2OVER OR R> OR 0= UNTIL ;
: (UQ.) <#Q #QS Q#> ;
: UQ. (UQ.) TYPE SPACE ;
: (Q.) DUP >R QABS <#Q #QS R> SIGN Q#> ;
: Q. (Q.) TYPE SPACE ;
: Q.R >R (Q.) R> OVER - SPACES TYPE ;
-->
```

```
A-QUADRUPLE.FTH / SCRN# 5
\ *Q QU/MOD Multiplication et division signees
: *Q DUP 3 PICK XOR >R DABS 2SWAP DABS UQ* R> ?QNEGATE ;
: QU/MOD >R >R DUP >R QABS R> R> R> 2DUP >R >R DUP >R DABS
        ROT R@ XOR >R UQ/MOD 2DUP OR 0=
        IF R> 0< IF >R >R QNEGATE R> R> THEN
                      R> R> R> 2DROP DROP
      ELSE R> 0< IF >R >R QNEGATE -1 -1 -1 -1 Q+ R> R>
                      R> 0< IF R> R> D+
                          ELSE R> R> 2SWAP DNEGATE D+ THEN
               ELSE R> 0< IF DNEGATE THEN R> R> 2DROP
      THEN THEN ;
: Q/    QU/MOD 2DROP ;
: Q/MOD QU/MOD 2ROT 2ROT 4DROP ;
: 4DUP 2OVER 2OVER ;
: 4SWAP 2ROT >R >R 2ROT R> R> ;
-->
```

```
A-QUADRUPLE.FTH / SCRN# 6
\ Comparaisons doubles et quadruples
: 4OVER >R >R >R >R 4DUP R> R> R> R> 4SWAP ;
: 8DUP 4OVER 4OVER ;
: D- DNEGATE D+ ;       : Q- QNEGATE Q+ ;
: D0= OR 0= ;           : Q0= OR OR OR 0= ;
: D= D- D0= ;           : Q= Q- Q0= ;
: D<> D= NOT ;          : Q<> Q= NOT ;
: DU< ROT SWAP 2DUP U< IF 2DROP 2DROP TRUE
     ELSE <> IF 2DROP FALSE ELSE U< THEN THEN ;
: QU< 2ROT 2SWAP 4DUP DU< IF 4DROP 4DROP TRUE
     ELSE D<> IF 4DROP FALSE ELSE DU< THEN THEN ;
: D<  2 PICK OVER = IF DU< ELSE NIP ROT DROP < THEN ;
: Q< 5 PICK 5 PICK 2OVER D= IF QU< ELSE 2SWAP 2DROP
     2ROT 2DROP D< THEN ;
: D> 2SWAP D< ;         : Q> 4SWAP Q< ;
-->
```

```
A-QUADRUPLE.FTH / SCRN# 7
\ 2@,2!,4@,4!,xconstant xvariable,D+!,Q+!.....
: DMIN 4DUP D> IF 2SWAP THEN 2DROP ;
: QMIN 8DUP Q> IF 4SWAP THEN 4DROP ;
: DMAX 4DUP D< IF 2SWAP THEN 2DROP ;
: QMAX 8DUP Q< IF 4SWAP THEN 4DROP ;
: 2@ DUP @ SWAP 2+ @ SWAP ;
: 4@ DUP 2@ ROT 4 + 2@ ;
: 2! DUP -ROT ! 2+ ! ;
: 4! >R 2SWAP R@ 2! R> 4 + 2! ;
: 2CONSTANT CREATE , ,      DOES> 2@ ;
: 4CONSTANT CREATE 2SWAP , , , , DOES> 4@ ;
: 2VARIABLE 0 0     2CONSTANT DOES> ;
: 4VARIABLE 0 0 0 0 4CONSTANT DOES> ;
: D+! DUP >R 2@ D+ R> 2! ;
: Q+! DUP >R 4@ Q+ R> 4! ;
-->
```

```
A-QUADRUPLE.FTH / SCRN# 8
\ Modification de NUMBER
VARIABLE DPL2
: CONVERT2 BEGIN 1+ DUP >R C@ BASE @ DIGIT WHILE
           0 2SWAP BASE @ 0 UQ* 2DROP 2ROT BASE @ 0 UQ* Q+
           DOUBLE? IF 1 DPL +! THEN R> REPEAT DROP R> ;
: (NUMBER?2) DPL2 OFF 0 0 ROT 0 0 ROT DUP 1+ C@
           ASCII - = DUP >R - -1 DPL !
           BEGIN CONVERT2 DUP C@ ASCII , ASCII / BETWEEN WHILE
           DUP C@ DPL2 @ MAX DPL2 ! 0 DPL ! REPEAT
           R> IF >R QNEGATE R> THEN C@ BL = ;
: NUMBER?2 FALSE OVER COUNT BOUNDS 2DUP >
           IF DO I C@ BASE @ DIGIT NIP IF DROP TRUE LEAVE
           THEN LOOP THEN
           IF (NUMBER?2) ELSE DROP 0 0 0 0 FALSE THEN ;

-->
```

```
A-QUADRUPLE.FTH / SCRN# 9
\ Modification de NUMBER
: (NUMBER2) NUMBER?2 NOT ?MISSING DOUBLE? DPL2 @ ASCII / = AND
            IF STATE @ IF 2SWAP [COMPILE] DLITERAL THEN
            ELSE 2DROP THEN ;


' (NUMBER2) IS NUMBER

ONLY FORTH DEFINITIONS DECIMAL
```

suite de la page 9

mais le PC, puisque PC il y a, et Forth, puisque esprit il est, ne peuvent raisonnablement cohabiter que dans quelque-chose qui ressemble un peu à TURBO-FORTH!

Vous avez dit standard? Quel standard?

...bien Forth à vous, Michel ZUPAN, Mars 88
67800 HOENHEIM

COURRIER: J'ai déja apprécié le F83, puis le TURBO-Forth (évaluation) et c'est avec joie et impatience que je vous écris aujourd'hui pour vous demander de bien vouloir m'envoyer la version définitive de TURBO-Forth (M1+M2+M3)...

TURBO-Forth est vraiment formidable. L'objection selon laquelle il était plus aisé de mettre au point en compilant bloc par bloc n'est pas réelle. J'ai moi-même expérimenté la possibilité de compiler un gros programme sous la forme de petits fichiers ASCII que j'ai ensuite rassemblés lorsque tout était au point.

André CHERAMY - 75005 PARIS

COURRIER: LES PROGRAMMES ECRITS EN FORTH

On me demande parfois si FORTH est réellement exploité dans le domaine professionnel. Je réponds, oui! Beaucoup en automatisme industriel. Mais il existe aussi des logiciels prestigieux écrits en FORTH, dont:

VP-PLANNER (clône de LOTUS 123)
RAPIDfile EASYWRITER
NEON POSTSCRIPT

et d'autres moins connus:

DELTA DRAW GAME DESIGNER
EXPERT-2 STARFLIGHT
RACE CAR SIMULATOR
PRO 30 DELTA DRAW

Cette liste n'étant pas limitée. Je tiens à remercier Mr F.CANNARD pour la doc concernant le NOVIX NC4016 et Mr L.MOREAU pour les extraits d'articles concernant les produits logiciels créés en Forth.

Marc PETREMANN
93160 NOISY LE GRAND

FAN DE THOMSON: bien sûr, je renouvelle mon abonnement, enthousiasmé par votre ASSEMBLER 6809. Mais mon point de vue va probablement différer du vôtre: je ne me soucie plus de professionnel, le PC m'importe peu.

Seul m'importe FORTH über alles, mais aussi quelque chose comme un LOGO écrit dans FORTH, peut-être FORTHlog.

Votre n° 41 confirme ma vision: FORTH sera pour la galerie ce que le VLSI a été en son temps. Mais FORTH appelle en complément une libération des contraintes de réservation. LOGO le langage des petits du primaire devrait être celui des bacheliers techniques.

FORTHlog semble être libéré des réservations, mais comment aborder son utilisation, quel manuel en constitue une bonne approche? PROLOG est-il un cousin?

A noter que votre ASSEMBLER m'a surpris dans la logique

suite page 14

volksFORTH-83 FORTH-Gesellschaft eV (c) 1985/86 we/bp/re/ks EXTEND.SCR Seite 1

1

```
\s                                                      14/12/87 YS
 Cher Secretaire,

        Je vous fais parvenir un mot qui, je l'espere, sera
utile aux utilisateurs de VFORTH-83 ne possedant qu un lecteur
de disquette. C'est comme un VIEW, mais ca ne lance pas
l'editeur. Utilisation:

WHEREIS TRUCMUCHE

Reponse:
is in TRUC.SCR Screen 1.

On se rapporte alors a son listing papier.
```

2

```
                                                        14/12/87 YS
Onlyforth definitions dos also
: whereis ( -- )
  ' >name 4- @ ?dup 0= abort" hand-made"
  base @ >r decimal
  ." is in "
  $200 u/mod file-link
  BEGIN @ dup WHILE 2dup fileno @ = UNTIL ( fcb )
  .file drop ."  Screen: " . r> base ! ;
```

3

```
\S                                                      14/12/87 YS
  L'ecran qui suit est l'implementation pour VFORTH-83 de la
structure CASE...OF...ENDOF...ENDCASE qui a mon avis permet
une "belle programmation". J'y ai rajoute l'ensemble

         { ....  }OF...ENDOF

ou les mots entre { et }OF evaluent une expression logique
partir du nombre sur la pile. Exemple dans l'ecran 5.
  Evidemment, les definitions de { et }OF peuvent etre reprises
telles quelles dans le FORTH-83 pour PC.


         Yves SURREL, Saint-Etienne
```

4

```
\ CASE OF { }OF ENDCASE                                 14/12/87 YS
variable csp
: case  sp@ csp ! ; immediate restrict
: of compile over compile = compile ?branch >mark
  compile drop ; immediate restrict
: endof compile branch >mark swap >resolve ; immediate restrict
: endcase compile drop
  BEGIN sp@ csp @ = not
  WHILE >resolve
  REPEAT ; immediate restrict
: { dup ;
: }of compile ?branch >mark compile drop ; immediate restrict
```

5

```
\ BETWEEN   Ex. pour CASE...OF...ENDOF...ENDCASE        14/12/87 YS
: between >r over > swap r> > or not ;
: .JOUR1&2  ( n -- )
  CASE
       1 OF ." lundi" ENDOF
       2 OF ." mardi" ENDOF
       { 1 7 between not }OF ." jour non valide" ENDOF
       ." ni lundi ni mardi"
  ENDCASE ;
```

0

suite de la page 12

de if..else..then; elle est inverse de celle du FORTH et c'est bien normal car vous êtes parti de beq, bne, etc... ne if..then correspond à bne..."then", mais ce n'est pas grave, le premier choc passé.

REPONSE: LOGO est un très beau langage, mais qui ne s'est pas démocratisé assez rapidement sur les micros. A l'heure où AMSTRAD annonçait son drive pour CPC 464 avec LOGO gratuit, la société ACT parlait de LOGO pour APPLE, IBM et autres à 2500 Fr minimum. Ils diffusaient même une version britannique pour SPECTRUM jamais importé en FRANCE. Puis ACT a été le moteur de l'ASSOCIATION AFUL (utilisateurs de LOGO) qui a fait un flop monstrueux. Passons sur le CENTRE MONDIAL de l'INFORMATIQUE qui n'a jamais publié le moindre article sur LOGO dans JEDI alors qu'une collaboration étroite nous avait été promise en son temps. Alors à qui la faute si LOGO s'éteint...

De toute façon, les machines et les logiciels disponibles deviennent si sophistiqués et offrent tellement de possibilité qu'il n'est plus possible de programmer soi-même des applications ambitieuses. Par contre, il devient stratégique de connaître les principes de base et la maipulation des logiciels majeurs: dBASE, MULTIPLAN, LOTUS, WORDSTAR, etc... et accessoire de savoir programmer. Dans un bureau équipé en matériel, on programme très peu, on exploite beaucoup les données et la communication. Et objectivement, quels sont les matériels les plus répandus: IBM et compatibles... Et je n'invente rien.

Alors si l'Education Nationale désire former des futurs utilisateurs de matériel bureautique, elle se doit d'utiliser ce qui se trouve sur le marché: un futur comptable a plus de chance de trouver un travail s'il a déja manipulé un tableur et un SGBD pultôt que LOGO ou LSE (soi-disant exigé en POLYTECHNIQUE, arghhh... chatouillez-moi!! - LSE c'est bien, c'est beau, mais c'est un autre flop...)

Maintenant, coté intérêt personnel et satisfaction de la curiosité intellectuelle, LOGO ou tout autre langage performant est bon à connaître. Il y a des notions communes qui vous profiterons le jour où vous changerez de système et de langage. Vous aurez moins de difficulté à passer de LOGO à PROLOG que de BASIC à n'importe quoi d'autre. Votre choix sera toujours le bon si vous y mettez la même passion à progresser que celle que vous avez mis à découvrir cet univers fantastique qu'est l'informatique. Il paraît qu'on exploite qu'une fraction de nos capacités intellectuelles; de même, on peut passer toute une vie à n'exploiter qu'une partie des possibilités de sa machine et pourtant, il y en a qui en changent tous les deux ans...

Concernant l'ASSEMBLER 6809 en FORTH que j'ai publié, j'ai remarqué la même chose que vous, mais il était trop tard pour corriger: la fonction similaire à bne est ne. Or, il faut inverser en utilisant eq. ne et eq sont des constantes, l'une étant paire, l'autre impaire. Donc, si dans un programme assembleur on trouve "bne", on remplace par eq if... then. Voilà, c'est pas compliqué.

LE SECRETAIRE

OCCAM

UN LANGAGE POUR LES ORDINATEURS DE LA PROCHAINE GENERATION

par M.DURANTON

Devant la demande sans cesse croissante de la part des utilisateurs de matériel informatique pour des machines dont les performances sont de plus en plus grandes, les constructeurs et les concepteurs sont amenés à réaliser des machines avec des architectures de plus en plus variées. Une voie est de créer une acrchitecture plus adaptée au traitement parrallèle. Mais une fois la machine réalisée, il faut pouvoir la programmer!

Or, si c'est (relativement) facile pour des machines faiblement parrallèles comme par exemple les CRAYs, il n'en est pas de même pour des ordinateurs composés de plusieurs dizaines, ou même des milliers de processeurs. On peut citer comme exemples de tel type de machines les ordinateurs de la série T de FPS, ou dans le cadre européen la famille des SUPERNODEs qui comprendra jusqu'à mille processeurs pour un même ordinateur.

Un des points communs des deux dernières machines est d'utiliser comme micro-processeur des TRANSPUTERs de la société INMOS, et comme langage de base OCCAM.

Ce n'est pas un hasard, car les TRANSPUTERs ont été conçus pour exécuter de manière optimale OCCAM. Le terme TRANSPUTER vient de TRANSistor et de comPUTER; c'est à dir que le TRANSPUTER doit être considéré pour les ordinateurs futurs comme le transistor pour les ordinateurs actuels; c'est à dire une brique de base.

Cependant, il ne faut pas considérer que la brique de base est peu puissante; le TRANSPUTER IMS T800 de INMOS est l'un des micro-processeurs les plus puissant actuellement; si ce n'est LE plus puissant. Il se compose, dans un seul circuit, d'un CPU 32 bits rapide, d'une unité de calcul flottant, de 4 Koctets de RAM, d'une interface mémoire permettant de connecter facilement n'importe quel type de boîtier mémoire, de bus de données et d'adresses sur 32 bits, de 4 liens de communication, d'un dispositif de gestion de processus, etc...

Pour les adeptes du FORTH, on peut dire que le CPU du TRANSPUTER ressemble un peu à la machine virtuelle Forth; il comporte une pile d'évaluation de trois registres, un pointeur d'instruction, un pointeur d'espace de travail et des pointeurs sur les listes des divers processus. A mon avis, il doit être relativement simple d'implémenter FORTH sur un TRANSPUTER, et pourquoi pas créer un langage FOCCAM qui soit un FORTH mais orienté gestion de tâches parrallèles; un mélange de FORTH et d'OCCAM... (Ndlr: TURBO-FOCCAM???...).

Nous reviendrons ultérieurement sur les liens de communication qui implémentent les canaux de communication OCCAM; disons tout de suite que c'est la manière privilégiée de connecter plusieurs TRANSPUTERs ensemble. Dans une version simple, un TRANSPUTER peut fonctionner sans circuit externe, chargeant son programme et communiquant par les liens. Dans une vesrsion plus complexe, chaque TRANSPUTER peut adresser 4 Goctets de mémoire!

Les performances du T800-30 sont les suivantes: 2.25 Mflops en représentation virgule floattant ANSI-IEE soit 6 MWhetstones par seconde en simple longueur pour ceux qui aiment les benchmarks. On peut dire que c'est environ 6 fois mieux que l'ensemble MC 68020 avec MC 68881 ou environ 3,5 fois mieux que l'ensemble 80386 avec 80387 tout en étant en un seul circuit (7 T800 offrent les perforamnces d'un CRAY 1S en mégaWhetstones).

Le TRANSPUTER s'accomode facilement des langages de haut niveau comme PASCAL, C et plus particulièrement OCCAM, créé antérieurement au TRANSPUTER pour gérer des processus parrallèles. Le code OCCAM est, sur le TRANSPUTER, aussi dense que celui du code assembleur écrit par un spécialiste (selon INMOS), donc le langage naturel de programmation du TRANSPUTER est OCCAM.

Après cette présentation du contexte d'utilisation d'OCCAM, voyons de plus près le langage. C'est un nouveau langage fondé sur le concept de parrallélisme et de communication, contrairement aux langages de programmation classiques qui sont séquentiels car ils sont prévus pour les machines classiques qui exécutent les actions les unes après les autres. OCCAM permet l'exécution de plusieurs processus concourrants pouvant se dérouler sur plusieurs processeurs. OCCAM découle des modèles PSP de HOARE et CCS de MILNER qui consacrent la

composition parrallèle comme méthode fondamentale de structuration en programmation.
La communication entre deux processus s'effectue par des entrées sorties synchrones à travers des canaux de communication. La sémantique formelle de CSP permet la lecture d'un programme OCCAM comme un prédicat formé d'un ensemble d'assertions dans une extension du calcul des prédicats où les règles de déduction peuvent être appliquées. On peut ainsi systématiquement utiliser des règles de transformation pour la construction correcte et l'optimisation des programmes.

OCCAM peut exprimer la structure hiérarchique et modulaire d'un système par encapsulation d'un ensemble de processus communiquants permettant à l'environnement de le voir comme un seul processus. Un canal peut être considéré comme un composant élémentaire de synchronisation entre deux processus. En OCCAM, les instructions élémentaires sont des processus et donc peuvent s'exécuter en parrallèle avec d'autres. Les constructeurs du langage permettent l'exécution parrallèle ou séquantielle d'un ou de plusieurs processus, ainsi qu'une sélection non déterministe d'un parmi un ensemble de processus, en fonction de conditions logiques appelées "gardes". Un répétiteur complète la liste des constructeurs du langage.

Un programme >OCCAM peut s'exécuter sur un processeur ou sur plusieurs; dans ce cas-là, celui-ci partage son temps entre les processus concurrents et des valeurs dans la mémoire réalisent des canaux. Les règles de transformation de programme permettent cette souplesse en altérant le degré de parralélisme et en le réalisant par quasi-parrallélisme sans changer la sémantique du programme. Le langage traite aussi bien les aspects systèmes tels que le traitement des interruptions et la multi-programmation.

Deux processus concurrents en OCCAM communiquent par un protocole de rendez-vous. Quand l'émetteur d'un message est prêt à transmettre vers un récepteur non encore prêt à recevoir, il va attendre que le récepteur indique la possibilité de recevoir le message. Quand le récepteur est prêt, le message est envoyé et les deux processus continueront dans des tâches indépendantes.

La syntaxe du langage OCCAM est sommairement décrite dans l'annexe: "OCCAM: une introdctio rapide".

Un exemple de programme OCCAM est donné dans l'annexe: "OCCAM: un exemple".

Cet exemple n'a aucune portée pratique: il s'agit de réaliser N processus parrallèles qui s'exécutent simultanément et qui ne font qu'envoyer leur numéro vers l'écran. Comme les communications sont point à point, on a besoin d'un processus (le "screen-mixer") qui affiche les numéros séquentiellement à l'écran sans favoriser un processus. Un autre problème est l'arrêt correct des processus sans "dead-lock". L'exemple présente une manière simple d'y arriver. Attention, l'exemple est donné pour voir la syntaxe OCCAM et quelques problèmes; il n'est pas du tout optimisé. On remarque cependant que l'indentation est essentielle en OCCAM; un processus peut se composer de sous-processus, mais alors ceux-ci sont indentés plus à droite. Par exemple:

```
SEQ
  proc1
  proc2
```

est le processus séquentiel composé de proc1 et puis de proc2. Par contre:

```
PAR
  SEQ
    proc1
    proc2
  proc3
```

est le processus parrallèle composé de deux sous-processus: un premier lui-même composé de proc1 et proc2, et le second constituant du processus parrallèle est proc3.

L'indentation en OCCAM est équivalente au BEGIN-END du PASCAL ou des { - } du C. La réalisation de ces indentations est facilitée par l'éditeur du système de développement OCCAM. C'est d'ailleurs un éditeur très agréable car il est hiérarchique: le programme est entré sous forme d'arbre et chacune des feuilles peut-être un fichier texte séparé. Par exemple, lorsqu'on entre dans un programme, on voit s'afficher: ...PROGRAM TOTO on place le curseur à cet endroit et on entre dans le programme; s'affiche alors le nom des différentes parties du programme. Si on se place sur une de ces parties, on peut entrer dedans et voir les en-têtes des procédures; puis on peut entrer dans la procédure pour voir les déclarations ou le code. De cette façon, on peut choisir à quel niveau on veut lire un programme, et on ne s'encombre pas de texte inutile. Dans le listage, qui est une mise "à plat" de la structure, un point d'entrée est donné par -- en début de ligne (ce qui correspond à une ligne de commentaire en OCCAM).

Il y a bien d'autres choses à dire sur OCCAM, mais mon but est seulement de présenter sommairement ce langage qui est promis à un certain avenir.

REFERENCES:

Manuel de programmation OCCAM - INMOS

Introduction à OCCAM - IMAG/LGI 1983

OCCAM:
UNE INTRODUCTION RAPIDE

INTRODUCTION:
OCCAM est un nouveau langage de programmation supportant des applications concurrentes dans lesquelles plusieurs parties d'un système fonctionnent séparément et inter-réagissent. Il peut être utilisé pour de nombreuses applications actuelles, particulièrement celles impliquant l'utilisation de microprocesseurs et de temps réel.

OCCAM sera essentiel pour les applications futures impliquant l'interraction de plusieurs centaines de composants de calcul.

DECLARATIONS:
La déclaration des variables peut se faire à n'importe quel moment dans un programme, tant que la variable est déclarée avant le processus qui l'utilise. Une variable déclarée dans un processus englobant est aussi connue dans les processus englobés.

Les canaux se déclarent suivant les mêmes règles que les variables.

Il est possible de déclarer des tableaux de variables ou de canaux.

Plusieurs types de constantes sont possibles. Exemple:

```
DEF maximum=10,minimum=0:
VAR volume:
CHAN ampli,hp:
SEQ
  -----------
  -----------
  CHAN toto:
  PAR
    ---------
```

PROCESSUS PRIMITIFS:
Il y a trois processus primitifs à partir desquels tous les autres sont construits:

- un processus d'entrée sur un canal de communication
  CANAL ? VARIABLE
- un processus de sortie sur un canal de communication
  CANAL ! EXPRESSION
- un processus d'affectation
  VARIABLE := EXPRESSION

(note: l'affectation peut se décrire en termes de communication:

```
  PAR
    CANAL ! EXPRESSION
    CANAL ? VARIABLE
)
```

CONSTRUCTEURS:
Les processus élémentaires sont associés par des constructeurs qui sont:

SEQ : les processus sont exécutés séquentiellement. Un processus commence lorsque le précédent est terminé.

PAR : les processus sont exécutés en parrallèle. Le processus PAR se termine lorsque tous ses constituants sont terminés.

ALT : le processus qui se déroule est celui dont la garde est vraie. Le processus global se termine lorsque le processus élu se termine. Exemple:

```
DEF minimum=0:
VAR volume:
SEQ
  volume:=0
  WHILE TRUE
    ALT
      louder ? ANY
        SEQ
          volume:=volume+1
          amplificateur ! volume
      (volume > minimum) & softer ? ANY
        SEQ
          volume:=volume-1
          amplificateur ! volume
```

(note: un ALT avec priorité existe: PRI ALT)

REPLICATEURS:
Un constructeur peut être multiplié par un réplicateur qui est e la forme:

```
variable=[<première valeur> FOR
         <nombre des répliactions>]
```

Pour exemple, le programme suivant en PASCAL:

```
FOR i:=1 TO 10 DO
BEGIN
  a:=a+i;
END;
```

s'écrit en OCCAM

```
SEQ i=[1 FOR 10]
  a:=a+i
```

Le réplicateur peut s'utiliser pour tous les constructeurs. Exemple: voici le programme OCCAM qui fait une estimation suivant la méthode de NEWTON-RAPHSON en utilisant une métode pipeline:

```
CHAN valeur[n+1]:
PAR i[0 FOR n]
  WHILE TRUE
    VAR x,estimation:
    SEQ
      valeurs[i] ? x
      valeurs[i] ? estimation
      valeurs[i+1] ! x
      valeurs[i+1] ! (estimation+(x/estimation))/2
```

STRUCTURES DE CONTROLE:
WHILE <condition> : répète un sous-processus tant que la condition est vraie.

```
IF  <condition1>
      <processus1>
    <condition2>
      <processus2>
    ...
```

Le IF OCCAM ressemble à un CASE en PASCAL: le premier processus dont la condition de garde est vraie s'exécute. Si aucun des processus n'est exécuté, le IF est équivalent à un NOP.

PROCESSUS NOMMES:
On peut créer des processus nommés avec passage de paramètres; ceux-ci peuvent être des variables, des canaux ou des valeurs. La syntaxe est la suivante:

```
PROC <nom> (VAR <ident>, CHAN <ident>, VALUE <ident>)=
  <corps du processus>:
```

DIVERS:
OCCAM dispose de nombreuses autres potentialités:

- accès à des tableaux par octets; exemple:

```
DEF alphabet="abcdefghijklmnopqrstuvwxyz":
SEQ i[1 FOR alphabet[BYTE 0]]
  lettres ! alphabet[BYTE i]
```

- gestion du temps comme dans l'exemple suivant:

```
DEF timeout =100:
VAR clock,x:
SEQ
  clock:=NOW
  ALT
    c1 ? x
      c2 ! ok.message;x
    WAIT NOW AFTER clock+timeout
      c2 ! timeout.message
```

- opérateurs complets:

```
+ * /\ \/ >< AND OR NOT TRUE FALSE << >>
+ - * / \
< > <= >= = <>
```

- etc...

ANNEXE: OCCAM UN EXEMPLE

```
-- PROGRAM ESSAI
-- essai
-- constantes
DEF nbprocess = 10 :
DEF EndBuffer = -2 :
DEF CR = #0D :
DEF LF = #0A :
DEF fin = -1 :

DEF texte = "Arrêt total de tous les processus..."

CHAN screen AT 1 :
CHAN keyboard AT 2 :
CHAN ecr [ nbprocess ] :
CHAN arret [ nbprocess ] :

-- screen.mix
PROC screen.mix ( CHAN scr[] ) =

  VAR fini,car,go :
  SEQ
    fini:=0
    go:=TRUE
    WHILE go
      ALT i=[0 FOR nbprocess]
        scr[i] ? car
          SEQ
            IF
              car=fin
                SEQ
                  fini:=fini+1
                  IF
                    fini=nbprocess
                      go=FALSE
                    TRUE
                      SKIP
                  TRUE
                    screen ! car :
```

```
-- process
PROC process ( chan ecr[] , arret [] ) =
  PAR i= [0 FOR nbprocess ]
    VAR notfini :
    SEQ
      notfini := TRUE
      WHILE notfini
        PRI ALT
          arret [ i ] ? ANY
            notfini := FALSE
          notfini & SKIP
            ecr [ i ] ! i + '0'
      ecr [i] ! fin :

-- c'est fini
PROC cestfini ( CHAN entree , halt [ ] ) =
  SEQ
    entree ? ANY
    PAR i = [ 0 FOR nbprocess ]
      HALT [ i ] ! ANY :
```

```
-- Voici enfin le programme principal
SEQ
  PAR
    cestfini ( keyboard , arret )
    process ( ecr , arret )
    screen.mix ( ecr )
  screen ! CR ; LF ; EndBuffer
  SEQ i = [ 1 FOR texte [ BYTE 0 ] ]
    screen ! texte [ BYTE i ]
  screen ! CR ; LF ; EndBuffer
```

PROLOG

CALCUL DE RESISTANCES

REMARQUE IMPORTANTE: ce programme est disponible en téléchargement ASCII ou KERMIT sur 3615 SAM*JEDI.

```
/***********************************************************************/
/*                                                                     */
/*               ** ** ** *************** ** ****                      */
/*               ** ** ** *************** ** ****                      */
/*      ********** ** ** * RESISTANCE * ** *************               */
/*      *        ** ** ** *************** ** ****        *             */
/*      *        ** ** ** *************** ** ****        *             */
/*      *                                                *             */
/***********************************************************************/
/*      Ce petit programme cherche et trouve a la place de l'electronicien */
/*  debutant l'assemblage de resistance standard donnant une resistance de */
/*  valeur non standard.                                                   */
/***********************************************************************/

predicates

     debut
     resultat(real)
     resistance(real)
     serie(real,real,real)
     parallele(real,real,real)

goal
     debut.

clauses
     debut :-
          write("Entrez la valeur de la résistance recherchée: "),
          readreal(X),resultat(X),nl,nl,debut.
     resultat(X) :-
          resistance(X), write("La résistance: ",X
          ," existe dans la serie "),nl.
     resultat(X) :-
          X=C,serie(A,B,C), write("La résistance: ",X
          ," s'obtient en mettant en serie deux résitances de: "
          ,A," et ",B," Ohm  "),nl.
     resultat(X) :-
          X=C,parallele(A,B,C), write("La résistance: ",X
          ," s'obtient en mettant en parallèle deux résitances de: "
          ,A," et ",B," Ohm  "),nl.
     resultat(X) :-
          not(resistance(X)),
          write(" pas de solution avec la serie de résistance utilisée"),
          nl,write(" utilisez une résistance ajustable. "),nl
          ,write("ou une résistance ayant une valeur approchée"),nl.

/***********************************************************************/
/* cette serie de resistance est standard, elle peut etre remplacé par le */
/* jeu en votre possession.                                               */
/***********************************************************************/

 resistance(1).   resistance(2).    resistance(5).     resistance(8).
 resistance(10).  resistance(220).  resistance(4700).  resistance(100000).
 resistance(12).  resistance(270).  resistance(5600).  resistance(150000).
 resistance(15).  resistance(330).  resistance(6800).  resistance(180000).
 resistance(18).  resistance(390).  resistance(8200).  resistance(220000).
 resistance(22).  resistance(470).  resistance(10000). resistance(270000).
 resistance(27).  resistance(560).  resistance(12000). resistance(330000).
 resistance(33).  resistance(680).  resistance(15000). resistance(390000).
 resistance(39).  resistance(820).  resistance(18000). resistance(470000).
```

```
resistance(47).  resistance(1000). resistance(22000). resistance(560000).
resistance(56).  resistance(1200). resistance(27000). resistance(680000).
resistance(68).  resistance(1500). resistance(33000). resistance(820000).
resistance(82).  resistance(1800). resistance(39000). resistance(1000000).
resistance(100). resistance(2200). resistance(47000). resistance(1500000).
resistance(120). resistance(2700). resistance(56000). resistance(1800000).
resistance(150). resistance(3300). resistance(68000). resistance(2200000).
resistance(180). resistance(3900). resistance(82000).

   serie(A,B,C) :-
       resistance(A),resistance(B), C = A+B.
   parallele(A,B,C) :-
       resistance(A),resistance(B),resistance(C),A = B , A = 2 * C.
   parallele(A,B,C) :-
       resistance(A),resistance(B),A () B, C = ((A*B)/(A+B)).

/* nicolas brun avril 1988 */
```

MODIFICATIONS DE META.FTH ET KERNEL.FTH DU LANGAGE TURBO-FORTH 83-STANDARD

Liste des modifications à apporter au fichier META.FTH du module M2 de TURBO.FORTH 83 Standard pour méta-compiler en extra-segment:

- suppression de TARGET-ORIGIN et THERE
- création de LCSET et redéfinition des mots C@-T à CSET-T

```
\ Mots d'accès mémoire extra-segment dans la cible
\ VARIABLE SEG-T VARIABLE DP-T
\ segment et offset de la cible
CODE LCSET ( oct seg adr -- )
     DS CX MOV   BX POP   DS POP   AX POP   AL 0 [BX] OR
     CX DS MOV   NEXT END-CODE
: C@-T    (S taddr -- char )    SEG-T @ SWAP LC@ ;
: @-T     (S taddr -- n )       SEG-T @ SWAP L@  ;
: C!-T    (S char taddr -- )    SEG-T @ SWAP LC! ;
: !-T     (S n taddr -- )       SEG-T @ SWAP L!  ;
: CSET-T  (S o taddr -- )       SEG-T @ SWAP LCSET ;
( le reste est inchangé de HERE-T à S,-T)
```

- supprimer les THERE et remplacer THERE CSET par CSET-T

- modification de la troisième ligne de HEADER:

```
: HEADER   (S -- )
     BL WORD C@ 1+ WIDTH @ MIN   ?DUP IF
     ALIGN   0 ,-T   ( view field )
     HERE CURRENT-T @ HASH DUP @-T ,-T
     HERE-T 2- SWAP !-T
     HERE-T HERE ROT S,-T   ALIGN   DUP LAST-T !
     128 SWAP CSET-T   128 HERE-T 1- CSET-T   THEN    ;
```

- modification de la définition de STATISTIQUES:

```
: STATISTIQUES
CR ." REFERENCES NON RESOLUES:" CR .UNRESOLVED CR
CR ."    STATISTIQUES: "
CR ." Segment hôte:               "  DSEGMENT U.
CR ." Dernière adresse hôte:      "  [FORTH] HERE U.
CR ." Segment cible:              "  SEG-T @ U.
CR ." Première adresse cible:     "  META 256  U.
CR ." Dernière adresse cible:     "  META HERE-T  U.
CR CR  ;
```

Liste des modifications à apporter au fichier KERNEL.FTH du module M2 de TURBO.FORTH 83 Standard pour méta-compiler en extra-segment:

- changer les déclarations d'adresses après avoir fait un ALLOC de 2*64K :

```
HEIGHT 2* ALLOC DROP  \ réserve  64K supplémentaires pour
la cible
DSEGMENT HEIGHT + SEG-T !    256 DP-T !     IN-META
```

- supprimer encore les THERE (à la fin)

- la seule définition à modifier est celle de CONTEXT:

```
VARIABLE  CONTEXT
     0 , 0 , 0 , 0 , 0 , 0 , 0 ,
( cette  suite est  plus simple qu'un ALLOT et un 0 LFILL
pour ERASE )
( 7 fois suffisent : VARIABLE en donne déjà un ...)
```

- La sauvegarde finale s'écrit:

```
META SEG-T @ 256 ( adr début)   HERE-T   ( adr fin)
      ONLY FORTH ALSO LSAVE  TEST.COM   FORTH
```

Petites modifications des opérateurs extra-segment

L'instruction 8086 POP DS est valide ! (code 1F): remplacer les DX POP DX DS MOV par DS POP :

```
CR .( Opérateurs mémoire 8 et 16 bits inter-segments)
CODE L!   ( val seg2 adr2  -- )
     DS AX MOV   BX POP   DS POP   0 [BX] POP
     AX DS MOV  NEXT END-CODE
CODE LC!  ( oct seg2 adr2 -- )
     DS CX MOV   BX POP   DS POP   AX POP   AL 0 [BX] MOV
     CX DS MOV   NEXT END-CODE
CODE L@   ( seg2 adr2 -- val )
     DS CX MOV   BX POP   DS POP
     0 [BX] PUSH   CX DS MOV   NEXT END-CODE
CODE LC@  ( seg2 adr2 -- oct )
     DS CX MOV   BX POP   DS POP   AX AX XOR
     0 [BX] AL MOV  CX DS MOV  1PUSH END-CODE

CR .( Mouvements blocs de mémoire inter-segments)
CODE LCMOVE (S from-seg from-adr to-seg to-adr count -- )
     CLD   IP BX MOV   CX POP   DI POP   ES POP
     DS DX MOV   IP POP   DS POP
     REP   BYTE MOVS   BX IP MOV
     DX DS MOV   NEXT END-CODE
CODE LCMOVE> (S from-seg from-adr to-seg to-adr count --)
     STD   IP BX MOV   CX POP   DI POP   ES POP
     CX DEC   DS DX MOV   IP POP   DS POP
     CX DI ADD   CX IP ADD   CX INC
     REP   BYTE MOVS   BX IP MOV   DX DS MOV
     CLD   NEXT END-CODE
```

Autres modifications sur le même principe :

```
CODE LFILL   (S seg adr lgr b -- )
     CLD   AX POP   CX POP   DI POP   ES POP
     REP   AL STOS   NEXT END-CODE
```

(il est inutile de sauver ES: ce registre est souvent utilisé sans être préservé comme la plupart des registres 8086 dans Forth)

```
CODE (GET) ( segm buffer len hndl -- len fl )
     DS AX MOV   ES AX MOV   BX POP   CX POP
     DX POP   DS POP   3F # AH MOV   21 INT
     AX PUSH   ES BX MOV   BX DS MOV
     U)=   IF   0 # AX MOV   THEN 1PUSH   END-CODE
CODE (PUT) ( segm buffer len hndl -- len fl )
```

```
DS AX MOV   ES AX MOV   BX POP   CX POP
DX POP   DS POP   40 # AH MOV   21 INT
AX PUSH    ES BX MOV    BX DS MOV
U>= IF    0 # AX MOV   THEN 1PUSH  END-CODE
```

Petite modification de DIR:

- ajout d'un UPC dans (DIR) (si i:*.* minuscules)

```
: (DIR) ( string -- )
   128 SET-DMA ( valeur habituelle DTA in PSP )  BASE @
>R DRV >R DECIMAL
   OVER 1+ C@ ASCII : =
   IF    OVER C@ UPC ASCII A - SELECT THEN
   PAD PLACE 0 PAD COUNT + C! PAD 1+ 16 (SEARCH0)
   IF   CR
        BEGIN .NAME #OUT @ 70 > IF CR THEN
              (SEARCH) NOT
        UNTIL
   THEN #OUT @ IF CR THEN
   DRV ASCII A + EMIT ." : "  FREE D.
   ." octets libres "
   R> SELECT R> BASE ! ;
```

REMARQUE IMPORTANTE: ces modifications n'influencent pas la syntaxe ou le comportement des versions modifiées par rapport aux précédentes versions. Elles apportent seulement une meilleure qualité de fonctionnement.

ERREURS DANS LE SOURCE DE TURBO-FORTH 83-STANDARD

Mr Charles MOHR nous signale une erreur dans la définition du mot SEARCH. Remplacer le mot U>= par U<=

```
: SEARCH   ( stradr strlen badr blen --- n f)
    FOUND OFF   SWAP >R  2DUP U<=
    IF   OVER - 1+ 2 PICK C@ R@ -ROT >R
         BEGIN R@ SCAN-1ST DUP
               IF   >R 3DUP SWAP COMPARE 0=
                    IF   FOUND ON R> DROP 0 >R   THEN
                    R>
               THEN  DUP
         WHILE 1 /STRING
         REPEAT R>  2DROP  -ROT
    THEN  2DROP R> - FOUND @  ;
```

TELEMATIQUE

JEDI SUR 3615

par Marc PETREMANN

J'avais proposé aux adhérents, voici déjà quelques mois, de rechercher un prestataire de service pour héberger un service télématique JEDI. Comme personne ne s'est manifesté avec vigueur, en dehors de deux propositions de dépannage (Mr FORGET et Mr CHANDRU), j'ai encore été obligé de tout faire moi-même.

Un contact a été pris avec la Sté VICTEL et l'idée d'héberger JEDI les a intéressé. Manque de bol, le code JEDI est déjà pris par l'évènement du JEUDI (eh, oui, ils ont des lecteurs faibles en 'aurtografe' qui tapent JEDI pour JEUDI; et JEUDI est une marque déposée depuis le quatrième jour de la création....). Donc, VICTEL s'occupe de SAM et le code d'accès au service télématique de JEDI est: SAM*JEDI via le 3615.

Bien sûr, j'attends les critiques: encore le 3615, ce n'est pas donné! Mais VICTEL est une SARL qui ne vit pas que d'eau fraîche et de chansonnette. Ils acceptent de nous héberger sans garantie d'une rentabilité immédiate. Par contre, si les recettes deviennent significatives, il a été convenu du principe de reverser un pourcentage à JEDI.

La solution du prestataire de service nous dégage de tout souci matériel, logiciel, maintance et disponibilité. Par contre, à nous d'animer ce service.

Tout de suite et dès maintenant:

- téléchargement des programmes parus ou à paraître dans JEDI.

Très bientôt:

- FORUM de trucs, communication et BALs adhérents; possibilité d'assistance logicielle sur des questions de programmation.

Le service SAM*JEDI est évolutif. Si vous avez des suggestions n'hésitez pas.

LE TELECHARGEMENT:
Tout de suite et dès maintenant, téléchargez les programmes ASCII à l'aide d'un programme de capture TELETEL vers IBM (ou compIBM) et/ou ATARI et/ou APPLE (autres si compatibilité logicielle).

Téléchargez également en protocole KERMIT des logiciels exécutables (.COM ou .EXE) pour IBM (ou compIBM). Le logiciel KERMIT est disponible auprès de l'ASSOCIATION JEDI contre 10 timbres à 3,70 Fr ou 30 Fr de supplément à toute commande de logiciel ou revues JEDI (en dessous de 70 Fr de commandes, payez par timbres postaux, SVP).

LE CABLE DE LAISON MINITEL - MICRO:
Rabattez-vous sur vos revues habituelles. Elles ont pratiquement toutes diffusé le plan du câble de liaison MINITEL - MICRO.

LES CARTES:
Carte KORTEX, WINNTEL, etc.. seront les bienvenues.

LES LOGICIELS:
KXCOM pour carte KORTEX, LCECOM pour câble PC-MINITEL.

CE QUI EST DISPONIBLE SUR SAM*JEDI:

TURBO.COM, EDIT.COM, EDITERR.MSG, FORTH.VOC, ROOT.VOC	package TURBO-Forth, module 1
ERATHOS.FTH	benchmark crible d'Erathosthène
RESISTAN.PRO	calcul de resistances en PROLOG

Les fichiers sont identifiés de la manière suivante:

L A N G A G E S   PC Compat.

#	Programme	Type	Desciption
1	ERATHOS.FTH	J A	CRIBLE ERATHOSTHENE
3	TURBO.COM	J B	TURBO F83-STANDARD

etc...

- J signifie qu'il s'agit d'un programme d'origine JEDI.
- A signifie que le programme est au format ASCII, donc téléchargeable tel quel ou par KERMIT.
- B signifie que le programme est en binaire, donc téléchargeable seulement en binaire.

COMMENT SE CONNECTER SUR SAM*JEDI:
(pour les adhérents concernés)
(information FRANCE TELECOM)

- DEPUIS LES DOM:
  .Télétel 3   : 36 15

Composez ensuite le nom abrégé du service ou le 20 80 suivi du numéro d'identification du service (N° Transpac amputé de son premier chiffre). Exemples:

.en métropole   :   100600078
.au départ des DOM:2080 00600078

Attention, certains services qui, depuis la métropole sont accessibles par nom abrégé, ne le sont pas forcément depuis les DOM.
Si ce n'est pas précisé dans l'annuaire, contactez le

correspondant des usagers du service.

- DEPUIS L'ETRANGER:

Utilisation du réseau téléphonique international (pour Télétel 1, 2 et 3):

Télétel 1 : 33 36 43 13 13
Télétel 2 : 33 36 43 14 14
Télétel 3 : 33 36 43 15 15

L'annuaire électronique est accessible par Télétel 2 puis:

- code AE pour la version francaise
- code ED pour la version anglaise

Télétel peut être appelé directement si le service est raccordé au réseau téléphonique.

Utilisation des réseaux de commutation par paquets (pour Télétel 1 et 2):

Un certain nombre de pays ont un réseau commuté par paquets relié au NTI:

- soit par l'intermédiaire des points d'accès VIDEOTEX compatibles TELETEL (donc utilisation de mots abrégés possible).
- soit par des PAD ASCII X3.

Les terminaux utilisables sont, selon les cas, des Minitel ou micro-ordinateurs équipés de modems adéquats.

L'annuaire électronique est accessible:
- NUA 2080 35002330 pour la version francaise
- NUA 2080 3500233002 pour la version anglaise.

Le service GROOM (pour Télétel 3):

Raccordé à TRANSPAC, le service GROOM, accessible sur abonnement, permet à tout usager hors de France, d'accéder aux services kiosques de Télétel.

Cas particuliers: Télétel et les réseaux Vidéotex des autres pays. Entre la FRANCE et la RFA, l'accord passé entre les deux pays permet à un Minitel en France de consulter les services BILDSCHIRMTEXT en RFA, et réciproquement.

.en composant 36 22 49 49

Pour en savoir plus, la brochure "APPELEZ TELETEL DEPUIS LE MONDE ENTIER" (Francais et Anglais), précise entre autres pour chaque pays:

- quels sont les Minitel homologués
- les distributeurs de Minitel
- la consultation des services Télétel installés en France
- les passerelles possibles
- où consulter le service

Pour se procurer la brochure, écrivez à:
CNET PARIS A
Service de la Documentation Technique
38/40 avenue du Général-Leclerc
F - 92131 ISSY LES MOULINEAUX

TELEMATIQUE

CONNEXION AU RESEAU BTX (BILDSCHIRMTEXT) AVEC UN MINITEL

par Marc PETREMANN

Matériel: un minitel (un dictionnaire ALLEMAND-FRANCAIS)

Le système BILDSCHIRMTEXT équivaut au système TELETEL français pour la RFA (hallo, freunde...) ou PRESTEL pour la GRANDE-BRETAGNE (hello kids...). Il existe entre le BTX et TELETEL un pont: 36.22.49.49 (voir tarif via le 11 en FRANCE). Nous avons fait un petit essai:

```
Bildschirmtext

Deutsche Bundespost



15.04.88  13:52

Guten Tag

CNET
Passerelle Experimentale
```

Nous n'allons pas insister. Si votre PC est connecté à votre MINITEL et que vous pouvez utiliser indifférement votre clavier PC ou MINITEL pour répondre voici les manipulations à connaître:

*0 SUITE pour le menu général BTX

nn à sélectionner entre 00 et 99 sans SUITE pour sélectionner un choix dans le menu.

*n..n# pour sélectionner une page directement ou un service

équivaut à suite si l'affichage est reporté sur plusieurs pages consécutives.

ATTENTION: de nombreux services sont à accès codé, voire payant; depuis le menu général:

81 Dialogue à distance (pour germanophones...)
82 TELEX
83 TELE-ACHAT
84 BTX-International

71 Informations générales sur BTX

99 déconnexion à BTX

Voilà pour les premières informations concernant BTX. Si nous apprenons l'existence de services intéressants, nous ne manquerions pas de vous communiquer la marche à suivre.